NASA CR-188285

CSDL-T-1221

A SIX DEGREE OF FREEDOM, PLUME-FUEL OPTIMAL TRAJECTORY PLANNER FOR SPACECRAFT PROXIMITY OPERATIONS USING AN A* NODE SEARCH

by

Mark Charles Jackson

May 1994

Master of Science
Massachusetts Institute of Technology

(NASA-CR-188285) A SIX DEGREE OF FREEDOM, PLUME-FUEL OPTIMAL TRAJECTORY PLANNER FOR SPACECRAFT PROXIMITY OPERATIONS USING AN A* NODE SEARCH M.S. Thesis - MIT (Draper (Charles Stark) Lab.) 147 p

N94-32388

Unclas

G3/13 0010458

The Charles Stark Draper Laboratory, Inc.
555 Technology Square, Cambridge, Massachusetts 02139-3563

A Six Degree of Freedom, Plume-Fuel Optimal Trajectory Planner for Spacecraft Proximity Operations Using an A* Node Search

by

Mark Charles Jackson

B.S.E., United States Naval Academy
(May, 1982)

SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1994

Signature of Author Mark C Jackson
Department of Aeronautics and Astronautics
May, 1994

Approved by Paul A. DeBitetto
Paul A. De Bitetto
Technical Supervisor, Draper Laboratory

Certified by Wallace E Vander Velde
Professor Wallace E. Vander Velde
Thesis Supervisor

Accepted by _____
Professor Harold Y. Wachman
Chairman, Departmental Graduate Committee

A Six Degree of Freedom, Plume-Fuel Optimal Trajectory Planner for Spacecraft Proximity Operations Using an A* Node Search

by

Mark Charles Jackson

B.S.E., United States Naval Academy
(May, 1982)

Submitted to the Department of Aeronautics and Astronautics
on May 6, 1994 in partial fulfillment of the requirements for the
degree of Master of Science

Abstract

Spacecraft proximity operations are complicated by the fact that exhaust plume impingement from the reaction control jets of space vehicles can cause structural damage, contamination of sensitive arrays and instruments, or attitude misalignment during docking. The occurrence and effect of jet plume impingement can be reduced by planning approach trajectories with plume effects considered. An A* node search is used to find plume-fuel optimal trajectories through a discretized six dimensional attitude-translation space. A plume cost function which approximates jet plume iso-pressure envelopes is presented. The function is then applied to find relative costs for predictable "trajectory altering" firings and unpredictable "deadbanding" firings. Trajectory altering firings are calculated by running the spacecraft jet selection algorithm and summing the cost contribution from each jet fired. A "deadbanding effects" function is defined and integrated to determine the potential for deadbanding impingement along candidate trajectories. Plume costs are weighed against fuel costs in finding the optimal solution. A* convergence speed is improved by solving approach trajectory problems in reverse time. Results are obtained on a high fidelity Space Shuttle/Space Station simulation. Trajectory following is accomplished by a six degree of freedom autopilot. Trajectories planned with, and without, plume costs are compared in terms of force applied to the target structure.

Thesis Supervisor: Professor Wallace E. Vander Velde
Professor of Aeronautics and Astronautics

Technical Supervisor Paul A. De Bitetto
Charles Stark Draper Laboratory, Inc.

Acknowledgment

May 6, 1994

This thesis was prepared at the Charles Stark Draper Laboratory, Inc., under contract NAS9-18426.

Acknowledgments

First I would like to thank my Technical Supervisor, Paul De Bitetto for spending so many hours reading my thesis and reviewing it with me line-by-line. His advice has greatly improved the quality and usefulness of this document.

I would also like to thank my Thesis Supervisor, Professor Wallace Vander Velde for his careful reviews, constructive criticism, and expert technical guidance.

The people of the Draper Simulation Lab were extremely helpful, especially Linda Leonard who constantly looked after me and my accounts and provided expert guidance and a friendly face.

Most importantly, though, I would like to thank my wife Jana, without whom I could not have finished this thesis. During the last few months, she has not only supported me, she has carried me.

The sacrifices made by my family were summarized by the poignant question asked by Jamie, my four-year-old daughter, when she was told that her father had finished his thesis: "Does this mean you'll be a daddy again?"

Table of Contents

List of Figures

INTRODUCTION

CHAPTER ONE

Continued exploration and exploitation of space will bring about increased spacecraft operations in the proximity of large, complex structures in orbit. Close proximity operations are complicated by the fact that exhaust plumes from the reaction control jets of space vehicles can cause structural damage, contamination of sensitive arrays and instruments, or attitude misalignment during docking. For our purposes, close proximity operations will be defined as operations at ranges where plume impingement is a concern – inside about 500 feet for the Space Shuttle.

The occurrence and effect of jet plume impingement can be reduced by planning approach trajectories with plume effects considered. Trajectory selection impacts plume impingement in two ways. First, the final approach trajectory determines the geometric positioning of jet groups relative to target structures, thereby determining which jet groups are likely to cause damage or misalignment. Second, the trajectory choice determines how the vehicle is affected by orbital mechanics. Trajectories which start from ahead of the target in the orbital direction, for example, require more earthward firings to counter a "sagging" effect on the approach caused by slowing the vehicle's orbit. Thus, the commanded trajectory determines which jets will point at which target structure, and influences whether or not they will be needed. An intelligent trajectory planner can reduce plume costs by searching for trajectories which minimize the potential for impingement. Such a planner should consider both translation and attitude since together these determine the degree of plume impingement for a given jet firing.

There are several challenges to developing a successful trajectory planner for close-in proximity operations. First, the planner must have a means for determining a plume impingement cost. The cost should be expressed as a function of structural component locations within each jet exhaust flowfield, and should be further weighted by the sensitivity of each component to impingement. Plume impingement costs should be weighed against fuel usage to avoid fuel waste. Second, the planner must have a simplified model of the relevant translational and rotational mechanics associated with orbital proximity operations. A tradeoff exists between the complexity of the model and the computational efficiency of the algorithm. Finally, if the trajectory is to be followed by an automatic control system, the planner must take into account that disturbances and small variations in initial conditions make it very difficult to predict which jets will actually be needed at a particular point along the trajectory. Thus, the planner must penalize trajectories which have large numbers of jets pointed at sensitive structural components for long periods of time.

The approach taken in this thesis was to design a generic trajectory planner for spacecraft operating in close proximity to satellites of known structure. The mass properties of the spacecraft, its jet configuration, and the target satellite's structural composition, are read in at run time. The speed of the planner was considered important to increase its utility as a ground or space-based tool, but code optimization was done only on a limited basis. It is assumed that the trajectories will be followed by an automatic feedback controller.

PREVIOUS AND PROPOSED METHODS

The general area of proximity operations has received considerable attention over the last several decades. The translational dynamics of closely orbiting spacecraft were derived by Clohessey and Wiltshire in 1960[1] (although a similar result was published by Hill in 1878 in a paper on lunar theory). There has been considerable work done in the Soviet Union in the area of automatic docking and space assembly[2], and recent work includes an advanced, six degree of freedom autopilot for the Space Shuttle[3].

Several studies have focused on fuel optimal trajectory planning for proximity operations. Bergmann et al applied a gradient descent technique to trajectory optimization over fuel costs[4]. The principles of artificial potential fields have been applied to proximity trajectory generation[5]. Two recent studies used the A* (pronounced "A star") node search technique to find fuel optimal trajectories[6,7]. These works provide the basis for some of the A* strategies used here.

Concern over plume avoidance has increased significantly with the proposal of various space station configurations. Much effort has gone into accurate modeling of reaction jet plumes and their effects on target structures.[8] Several studies have considered "dynamic plume avoidance" – that is modifying the jet selection scheme to prevent or reduce plume impingement[9]. Dynamic plume avoidance can work well when alternate jet combinations are available to carry out a specific command. In many instances however, adherence to a trajectory requires firing a jet which will impinge the target. In such cases, we are left with a choice of accepting plume

impingement, or loss of control in one or more axes. It makes sense, therefore to try to plan the trajectory so as to minimize the need for such firings.

Several proposals have been made to include plume impingement in trajectory planning. Many of these involve modeling the plume as a truncated cone and applying collision detection algorithms[10]. The plume cost function used here is based on a simplified plume model which has several of the key characteristics of the actual jet plume. The cost due to pressure, heating and contamination, drops off as the square of the range – a much more realistic model than a solid cone which is truncated at a particular range from the jet. This function also approximates the "bulb" shape of the plume so that the cost decreases with angle off the plume axis. Chapter two presents the jet plume characteristics which are important and chapter three covers the plume cost model.

This thesis proposes using an A* node search technique, with a continuous plume cost function, to find plume-fuel optimal trajectories. The A* node search strategy of reference 11 is particularly well suited for problems of large dimension, because it uses heuristic knowledge (see chapter 2) of the problem to direct the search along the "most promising" path. A* is also well suited for complex cost functions, like the composite plume cost function presented in chapter three, the gradient of which cannot be conveniently expressed.

ASSUMPTIONS

The assumptions made during the trajectory planning phase are listed below. The spacecraft following the trajectory will hereafter be referred to as the *chase* vehicle and the target satellite as the *target*.

1) *The target is in a circular, or near circular orbit with known period and altitude.*

2) *The effects of disturbances such as gravity gradient torques and aerodynamic drag are small and can be ignored.*

3) *Spacecraft thrusters provide impulsive velocity increments.* This is a reasonable approximation when the time between firings is long in comparison with the jet on times.

4) *Spacecraft jet granularity is small and can be ignored.* This allows the propagation of states without running the spacecraft jet selection routine (see chapter 3).

5) *The effects of Euler coupling are small and can be ignored.* The attitude rates used in planning are on the order of 0.2 deg/s. The effects of Euler coupling for the 5-15 second sampling times considered are very small.

6) *The target satellite has a known structure and an attitude control system.* This enables us to determine the locations of target structural components within the local coordinate system.

7) *The time of flight from the start of the trajectory, to the goal is fixed.* This simplifies fuel cost estimation and the A* search framework. See chapter five for recommendations on perturbing time of flight from a nominal value to improve performance.

The target and chase vehicle (and thus the LVLH frame) are assumed to have an orbital rate of 0.066439 deg/sec. The mass properties of the Space Shuttle test vehicle vary with the test and are presented in chapter four.

Ultimately, it is hoped that an intelligent trajectory planner will become part of a larger proximity operations package for the Space Shuttle or other spacecraft, which will include automatic station keeping, maneuvers and docking. With that in mind, all trajectory testing was done with a six DOF feedback controller developed specifically for this project. The control details are not considered important to this thesis on except in so far as discussed in chapters three and four. The trajectories generated were tested on the Charles Stark Draper Laboratory's Interactive Controls and Displays Simulation (ICDS) which is a six DOF Space Shuttle Simulation with two rigid bodies.

Four chapters follow. Chapter two provides the background required to understand relevant translational and rotational mechanics as well as the

A* algorithm. Chapter three covers the application of A* to spacecraft trajectory generation and plume optimization. Chapter four provides the results of several simulation runs and chapter 5 has conclusions and recommendations for future work.

CHAPTER TWO

This chapter develops the fundamental ideas used in this trajectory generation application. The first three sections cover the coordinate systems and translational and rotational dynamics used to propagate a spacecraft's state relative to another orbiting body. These are followed by a discussion of the jet plume characteristics used in designing the plume impingement cost function of chapter three. Finally, the use of heuristics in the A* algorithm is explained and an example given to provide an understanding of the search technique used here to generate results.

Axis Systems and State Definition

Three coordinate frames are of interest here. The body reference frame, the local vertical local horizontal (LVLH) frame and the orbiter vehicle structural frame which is discussed in chapter 4. The body and LVLH frames we shall use here conform to NASA standards[12]. The body frame (Fig. 1a) is centered at the vehicle center of gravity, with the x, y and z axes defined with respect to the spacecraft structure in a right hand system. For the Space Shuttle the positive x axis passes through the nose, the positive y axis passes through the right wing, and the positive z axis passes through the bottom of the vehicle (Fig 1a).

The origin of the LVLH frame (Fig. 1b) is located at the target's center of gravity with the positive x axis in the direction of the orbital velocity, the positive z axis pointing toward the center of the Earth (or other attractive

body) and the plus *y* axis defined by the right-hand rule. Notice that this frame rotates once per orbital revolution relative to inertial space.

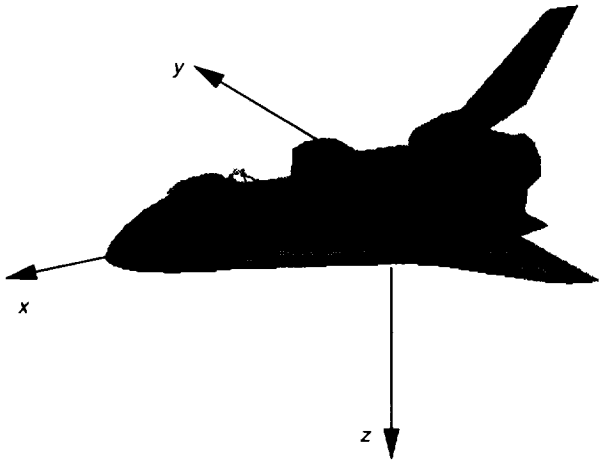


Figure 2-1a. Body axis system

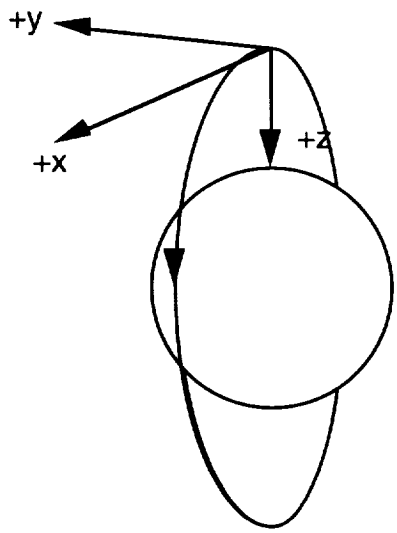
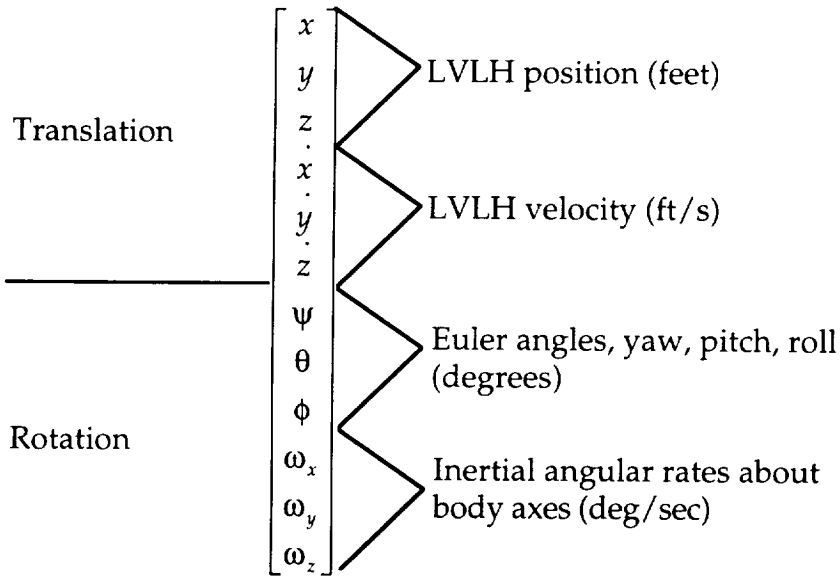


Figure 2-1b. LVLH coordinate frame.

We are interested in generating six dimensional trajectories – three translational states and three attitude states. Since most sensors will be designed to provide state information relative to a target structure, we will define our state vector in terms of the LVLH coordinate system (with the exception of angular rates which are inertial for reasons discussed later in this chapter). In order to propagate the state, we shall include the state derivatives in the state vector. The state vector used here is:



This vector uniquely describes the chase vehicle's linear and angular position and velocity with respect to the target.

The following sections develop the translational and rotational dynamics that apply to trajectory generation and jet plume impingement during proximity operations. The physical implications of the equations are examined to shed light on the plume impingement problem.

Translation

Reference 1 derives the linearized equations which govern the movement of two point masses in close *circular* orbits. These equations (1), known as the "Clohessey-Wiltshire equations" or "Hill's equations" describe the interaction between positions and velocities within an LVLH coordinate system centered at the target.

$$\ddot{x} = 2\omega_o \dot{z} + f_x \tag{2.1a}$$

$$\ddot{y} = -\omega_0^2 y + f_y \tag{2.1b}$$

$$\ddot{z} = -2\omega_0 \dot{x} + 3\omega_0^2 z + f_z \tag{2.1c}$$

where ω_0 is the orbital rate.

TRANSLATION WITHIN THE ORBITAL PLANE

We will consider the unforced case where $f_i = 0$. Equations (2.1a) and (2.1c) govern motion within (or parallel to) the orbital plane – or the x-z plane in LVLH coordinates. Note that in-plane motion is completely de coupled from out-of-plane motion as described by equation (2.1b). The x-z dynamics account for changes in the chase vehicle's orbit caused by movement within the plane.

To understand the in-plane interaction, consider a spacecraft in a perfectly circular orbit. A circular path is the orbit of minimum energy for a given minimum radius (**R** in Fig. 2-2). A spacecraft in a circular orbit has constant potential energy (altitude) and constant kinetic energy (tangential

velocity). If the spacecraft accelerates briefly in the orbital tangent direction (x_{LVLH}), its kinetic energy increases and its orbit changes shape to an ellipse, tangent to the original circle as shown in figure 2-2a. On this ellipse, the vehicle's kinetic energy is maximum at the point of tangency while its potential energy is maximum at the point of maximum altitude. If the acceleration is opposite the orbital direction, the ellipse of figure 2-2b results.

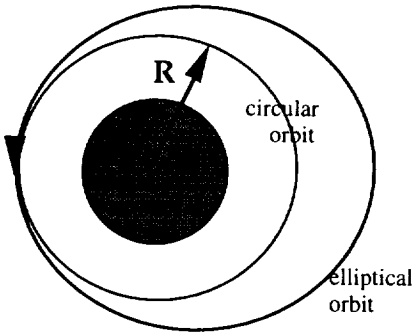


Fig 2a. Thrusting in orbital direction

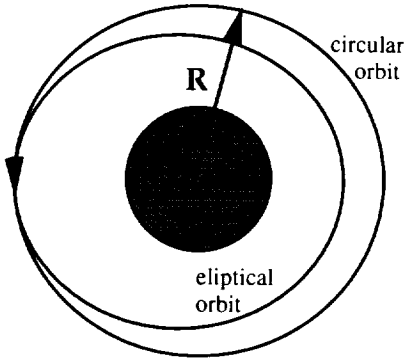


Fig. 2b. Thrusting opposite orbital direction

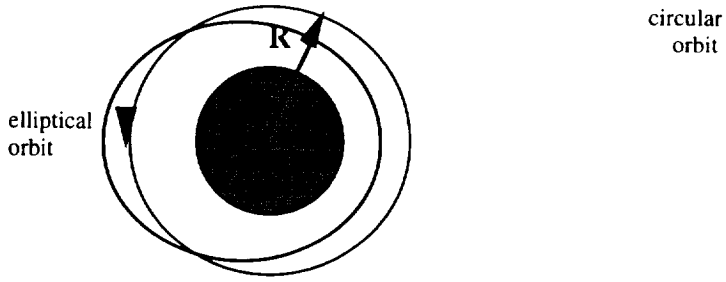


Fig. 2c. Displacement within the orbital plane

Figure 2-2. Effects of Energy Changes on Orbits

Now imagine a second spacecraft that is near the first when the in-plane acceleration occurs. The second spacecraft (the target) will see the first descend or climb as it enters its new orbit. This apparent motion is described by the first terms of equations (2.1a) and (2.1c).

Since the circular orbit has constant potential and kinetic energy, and since their sum is the minimum total energy for an orbit with minimum radius **R**, it follows that two spacecraft with different altitudes cannot be in the same orbit. If we displace an orbiting body in the LVLH positive z direction, while keeping its total energy constant, the ellipse of figure 2-2c results. The difference in vehicle altitudes is the z_{LVLH} coordinate, which is included in the second term of equation (2.1c). The interaction described by the two in-plane equations produces an apparent force toward the $(x\text{-}y)_{LVLH}$ plane when the chase is displaced with respect to the target (LVLH coordinate center) in altitude. A more precise description of this interaction is provided by the solution of the Clohessey-Wiltshire equations (equation 2.2 below).

EFFECTS OF IN-PLANE MECHANICS ON V-BAR AND R-BAR APPROACHES

The phenomena described above have a direct impact on proximity operations and plume impingement. Consider a spacecraft attempting a rendezvous with a target from ahead of the target in the orbital direction, positive x_{LVLH}. In the astronautics community, this direction is known as *V-BAR* (Fig. 2-3a). As the chase vehicle thrusts toward the target, its orbital velocity slows and it enters an elliptical orbit, inside the target's circular orbit, as described above. To stay on V-BAR, the chase vehicle must counter this "sagging" effect by occasionally firing earthward-facing thrusters. Similarly, if the chase approaches the target from behind, it will tend to "balloon" in altitude, requiring upward-facing thruster firings.

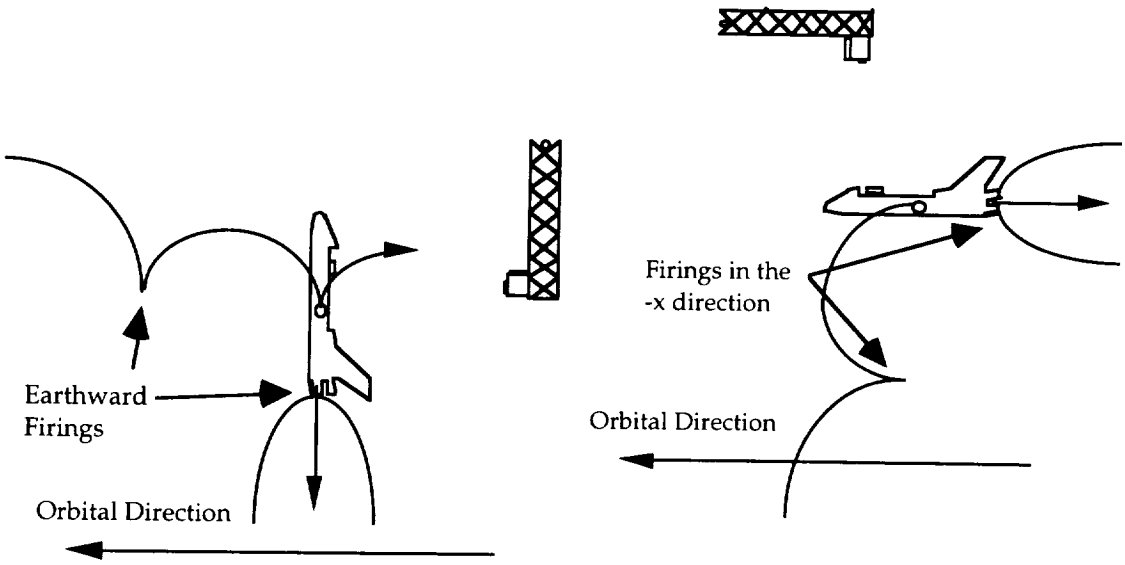


Figure 2-3a. V-BAR approach. Figure 2-3b. R-BAR approach
Figure 2-3. Typical docking approaches.

The LVLH positive z direction is known a *R-BAR*. Vehicles rendezvousing from below with the same tangential velocity as the target, will be drawn toward the target as race cars on a circular track are drawn together when the inside car accelerates to the outside car's speed. Jet firings in the positive and negative x directions then, can be used to control closure rates on R-BAR dockings. The coupling of the in-plane effects described above results in a requirement for repeated firings opposite the orbital direction during typical R-BAR approaches (Fig. 2-3b).

These in-plane dynamics significantly influence trajectory planning and plume impingement during proximity operations. If, for example, a V-BAR approach passes over sensitive solar arrays, the downward jet firings required to maintain altitude may cause excessive impingement.

OUT-OF-PLANE MECHANICS

The second of the Clohessey-Wiltshire equations (2.1b) is a simple second order differential equation whose solution is a sinusoid. The physical significance of this equation may be understood by imagining two orbiting spacecraft separated laterally by a small amount. Each orbit is a flat disk passing through the center of the Earth. These two disks intersect as shown in figure 2-4. From the figure, we can see that the orbital paths alternately converge and diverge as the spacecraft revolve. Therefore, any lateral displacement on one side of the earth results in an equal and opposite displacement on the other side. In the absence of control, a simple harmonic motion results.

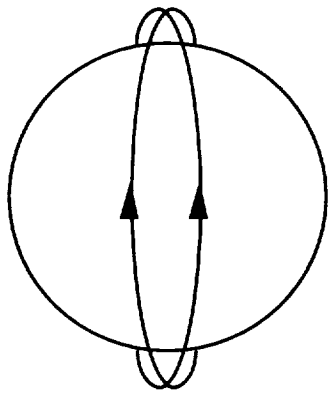


Figure 2-4.

Out-of-plane effects also influence which jets are fired and when. An approach from the positive y_{LVLH} direction, for example, may require thruster firings toward the target to reduce closure rates caused by these effects.

PROPAGATION OF THE TRANSLATIONAL STATE

The unforced solution of equations (2.1) can be expressed as a state transition matrix Φ such that:

$$\mathbf{x}(t+\Delta t) = \Phi(\Delta t)\mathbf{x}(t), \text{ where } \mathbf{x} = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T, \text{ and}$$

$$\Phi = \begin{bmatrix} 1 & 0 & 6(\omega_o\Delta t - \sin(\omega_o\Delta t)) & \frac{4}{\omega_o}\sin(\omega_o\Delta t) - 3\Delta t & 0 & \frac{2}{\omega_o}(1-\cos(\omega_o\Delta t)) \\ 0 & \cos(\omega_o\Delta t) & 0 & 0 & \frac{\sin(\omega_o\Delta t)}{\omega_o} & 0 \\ 0 & 0 & 4-3\cos(\omega_o\Delta t) & \frac{2}{\omega_o}(\cos(\omega_o\Delta t)-1) & 0 & \frac{\sin(\omega_o\Delta t)}{\omega_o} \\ 0 & 0 & 6\omega_o(1-\cos(\omega_o\Delta t)) & 4\cos(\omega_o\Delta t)-3 & 0 & 2\sin(\omega_o\Delta t) \\ 0 & -\omega_o\sin(\omega_o\Delta t) & 0 & 0 & \cos(\omega_o\Delta t) & 0 \\ 0 & 0 & 3\omega_o\sin(\omega_o\Delta t) & -2\sin(\omega_o\Delta t) & 0 & \cos(\omega_o\Delta t) \end{bmatrix}. \quad (2.2)$$

Equations (2.2) allow the state at time t+Δt to be determined from the state at time t. For trajectory planning, it will be useful to express the velocity required at time t = t_o in terms of the current position at t_o and the desired position at time $t_1 = t_o + \Delta t$. To this end, Φ can be partitioned as:

$$\Phi = \begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{bmatrix}.$$

Which allows us to solve for the velocity required at time t_o:

$$\mathbf{v}_{r0} = \begin{bmatrix} \dot{x}_0 \\ \dot{y}_0 \\ \dot{z}_0 \end{bmatrix} = \Phi_{12}^{-1}\left(\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} - \Phi_{11}\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} \right). \quad (2.3)$$

The change in velocity, $\Delta \mathbf{v}_o$, required to achieve the new position in Δt is simply the required velocity minus the current velocity: $\Delta \mathbf{v}_o = \mathbf{v}_{ro} - \mathbf{v}_o$.

Two things bear noting at this point. First, equations (2.2) and (2.3) do not provide enough degrees of freedom to stipulate both desired position and velocity at time t_1. This will become important when we consider constraints, such as docking constraints, which will impose velocity limits as well as position conditions at the goal. Second, we can propagate the state backward in time by using Φ^{-1}, which can be computed easily as: $\Phi^{-1}(\Delta t) = \Phi(-\Delta t)$. The inverse of any partition of Φ can also be computed simply by substituting $-\Delta t$ for Δt.

Rotation

Three of the most common methods for expressing a spacecraft's attitude with respect to a reference frame are, Euler angles, quaternions and transformation matrices. (Reference 14 cites several more methods.) Euler angles are often used to describe the spacecraft state while quaternions are commonly used to propagate the state. Transformation matrices are usually used to transform position or attitude information back and forth between reference frames. In this application, we are mainly interested in state description and propagation so we shall discuss Euler angles and quaternions in the next two sections.

EULER ANGLES

Figure 2-5 shows three consecutive rotations of an arbitrary orthonormal basis, *x-y-z*. The first rotation is about the z axis by an amount ψ. The second and third rotations are about the *y* and *x* axes by amounts θ and ϕ respectively. The orientation of the new frame, *x"-y"-z"* can be described by the vector [ψ θ ϕ]. Note that while this vector describes a unique orientation with respect to *x-y-z*, the vector itself is not unique. There are an infinite number of rotational combinations resulting in *x"-y"-z"*. Also note that the order of application is important. The vector [ϕ ψ θ] does not generally describe the same orientation as [ψ θ ϕ].

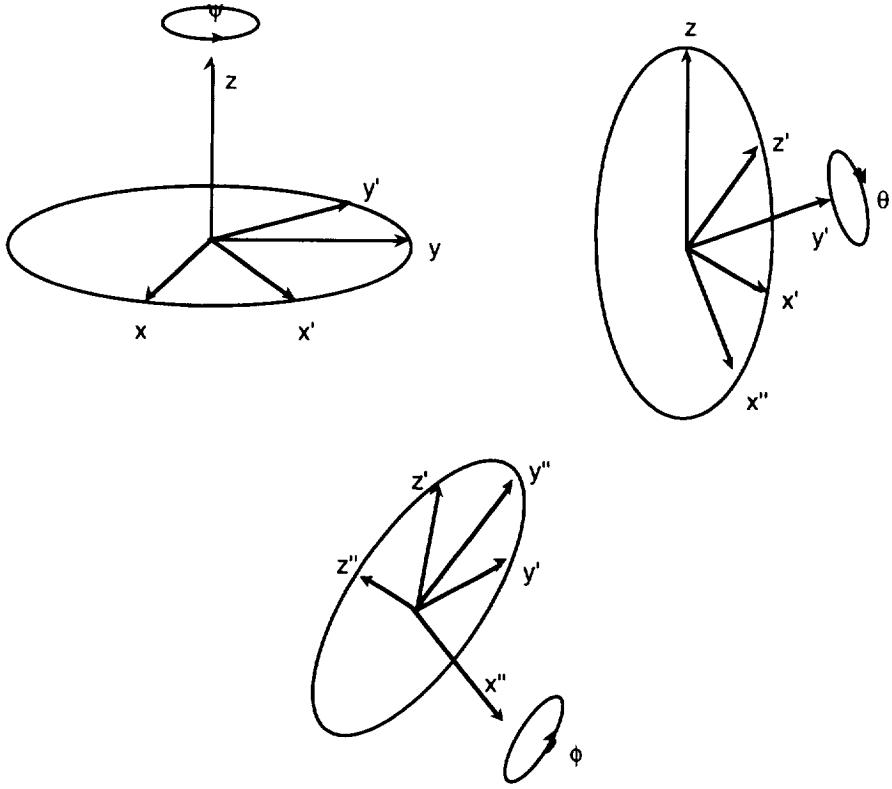


Figure 2-5.

When the sequence of rotations ψ, θ, and ϕ is used to describe the attitude of a spacecraft with respect to a reference frame such as the LVLH frame, these Euler angles are commonly named Yaw, Pitch and Roll.

QUATERNIONS: DEFINITION AND NOTATION

A quaternion **q** may be viewed as a complex number with three imaginary parts. Like ordinary complex numbers, quaternions consist of a real component and an imaginary component, but now the imaginary component is a three dimensional *vector*:

$$\mathbf{q} \equiv q_0 + iq_1 + jq_2 + kq_3 \tag{2.4}$$

where i, j and k are the unit vectors in an orthonormal system of imaginary axes. Figure 2-6 shows a quaternion with $q_3 = 0$ (in order to draw it).

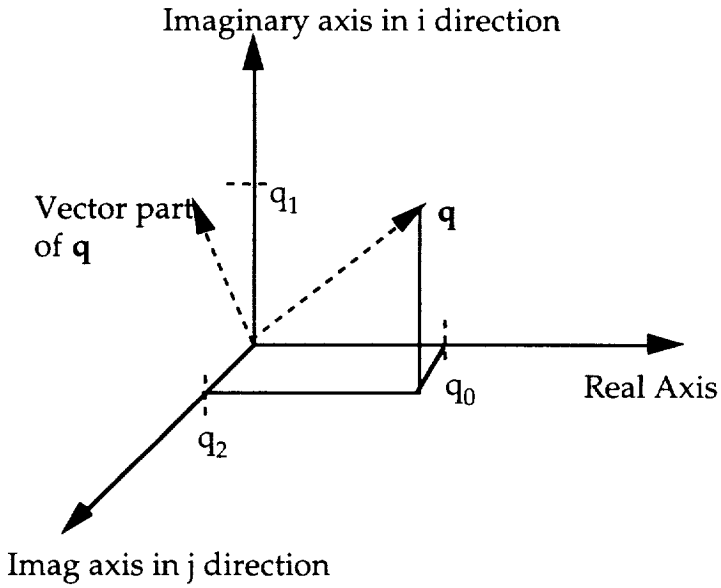


Figure 2-6. Real and imaginary parts of a quaternion with $q_3 = 0$.

The real part of a quaternion is called the scalar part and the imaginary part is called the vector part. We shall denote the scalar part of the quaternion $\mathbf{q}$, q_0 and the vector part $\vec{\mathbf{q}}$. Since i, j and k are orthonormal imaginary numbers, we can define their products to be:

$$\begin{aligned} i^2 &= j^2 = k^2 = -1 \\ ij &= -ji = k \\ jk &= -kj = i \\ ki &= -ik = j \end{aligned} \tag{2.5}$$

Using (2.4) and (2.5) we can find the product of two quaternions:

$$\begin{aligned} \mathbf{q}^a\mathbf{q}^b &= (q_0^a + iq_1^a + jq_2^a + kq_3^a)(q_0^b + iq_1^b + jq_2^b + kq_3^b) \\ &= (q_0^a q_0^b - q_1^a q_1^b - q_2^a q_2^b - q_3^a q_3^b) \\ &+ i(q_0^a q_1^b + q_1^a q_0^b + q_2^a q_3^b - q_3^a q_2^b) \\ &+ j(q_0^a q_2^b - q_1^a q_3^b + q_2^a q_0^b + q_3^a q_1^b) \\ &+ k(q_0^a q_3^b + q_1^a q_2^b - q_2^a q_1^b + q_3^a q_0^b) \end{aligned} \tag{2.6}$$

The conjugate of a quaternion is a straightforward extension of ordinary complex conjugation:

$$\mathbf{q}^* \equiv q_0 - iq_1 - jq_2 - kq_3 \tag{2.7}$$

Likewise, the magnitude of a quaternion comes from ordinary complex magnitudes:

$$\|\mathbf{q}\| \equiv \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \tag{2.8}$$

When $|\mathbf{q}| = 1$, $\mathbf{q}$ is said to be a *unit* quaternion.

It is common to arrange the four parameters of a quaternion into a column vector as: $\mathbf{q} = [q_0\ q_1\ q_2\ q_3]^T$. Three dimensional vectors can be

represented as purely imaginary quaternions with $q_0 = 0$. Thus a quaternion **r** represents a three dimensional vector if $\mathbf{r} = [0, \vec{\mathbf{r}}]$.

ROTATION OF VECTORS USING QUATERNIONS

Like transformation matrices, quaternions can be used to rotate three dimensional vectors in space. A vector $\vec{\mathbf{r}}$, represented as a purely imaginary quaternion, may be rotated by operating on it by a *unit* quaternion **q** as follows:

$$\mathbf{r}' = \mathbf{q}\mathbf{r}\mathbf{q}^* \tag{2.9}$$

It may be easily verified that the real part of $\mathbf{r}'$ is zero, so (2.9) maps purely imaginary quaternions (vectors) to purely imaginary quaternions. Expanding (2.9) by using (2.6) we can see that this mapping amounts to multiplying **r** by a matrix **D**:

$$\mathbf{D} = \begin{bmatrix} |\mathbf{q}|^2 & 0 & 0 & 0 \\ 0 & \left(q_0^2 + q_1^2 - q_2^2 - q_3^2\right) & 2\left(q_1 q_2 - q_0 q_3\right) & 2\left(q_1 q_3 + q_0 q_2\right) \\ 0 & 2\left(q_1 q_2 + q_0 q_3\right) & \left(q_0^2 - q_1^2 + q_2^2 - q_3^2\right) & 2\left(q_3 q_2 - q_0 q_1\right) \\ 0 & 2\left(q_1 q_3 - q_0 q_2\right) & 2\left(q_3 q_2 + q_0 q_1\right) & \left(q_0^2 - q_1^2 - q_2^2 + q_3^2\right) \end{bmatrix} \tag{2.10}$$

Closer inspection shows that the lower right 3×3 portion of **D** is of the same form as a direction cosine matrix when **q** has unit magnitude. (The upper left term has no effect on **r**, which has zero real part.) This confirms our assertion that (2.9) is a rotation, but, we still need to know how much, and in what direction $\vec{\mathbf{r}}$ is rotated.

Euler's Theorem of Rotation says that any rotation of a body, can be viewed as a rotation about a single axis **e**, the eigenaxis, by a specific amount, ϕ, the eigenangle. Thus **e** may be considered the direction of rotation and ϕ the magnitude. Since the transformation of (2.9) causes a rotation, there must be some relationship between **q** and the direction and magnitude of the rotation caused by **q**. We would like to express **q** in terms of **e** and ϕ. To this end, suppose we let:

$$\begin{aligned} \mathbf{q} &= \cos(\phi/2) + \sin(\phi/2)\mathbf{e} \\ &= \cos(\phi/2) + i e_1 \sin(\phi/2) + j e_2 \sin(\phi/2) + k e_3 \sin(\phi/2). \end{aligned} \tag{2.11}$$

Notice that the imaginary part of this **q** is just the eigenaxis scaled by $\sin(\phi/2)$. Carrying out the multiplications of (2.9), it may be verified that when $\mathbf{r} = [0, \vec{\mathbf{r}}]$, the resultant quaternion $\mathbf{r}'$ has zero real part and imaginary part equal to:

$$\vec{\mathbf{r}}' = \left(q_0^2 - \vec{\mathbf{q}} \cdot \vec{\mathbf{q}}\right)\vec{\mathbf{r}} + 2q_0\vec{\mathbf{q}} \times \vec{\mathbf{r}} + 2(\vec{\mathbf{q}} \cdot \vec{\mathbf{r}})\vec{\mathbf{q}} \tag{2.12}$$

To gain insight, let us consider the two extremes, $\phi = 0$ and $\phi = \pi$. When $\phi = 0$, the imaginary part of **q** is zero and $\mathbf{q} = [1\ 0\ 0\ 0]^T$ (by 2.11). Substituting this **q** into (2.12), we see that $\mathbf{r}'$ is simply $\vec{\mathbf{r}}$ – no rotation. At the other extreme, if $\phi = \pi$, then $q_0 = 0$, **q** is $[0\ e_1\ e_2\ e_3]$ and (2.12) becomes:

$$\vec{\mathbf{r}}' = -\vec{\mathbf{r}} + 2(\vec{\mathbf{r}} \bullet \mathbf{e})\mathbf{e}$$

Figure 2-7 shows that this is a *reflection* (180^o rotation) about the vector part of **q**, which is the eigenaxis, **e**.

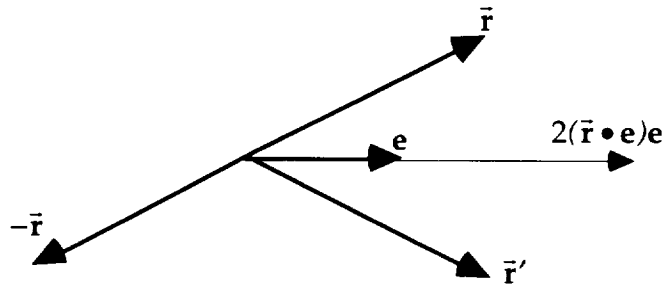


Figure 2-7. Reflection about the eigenaxis.

When ϕ is between 0 and π, the rotation magnitude is ϕ and its direction is **e**. (This may be rigorously verified by substituting (2.11) into (2.9) and showing that the result is the Rodriguez formula for rotating vectors – see reference 13). Thus, if we desire to rotate a three dimensional vector by an amount ϕ, about an axis, **e**, we should use $\mathbf{r}' = \mathbf{qrq}^*$ and let **q** have the form of (2.11).

Equation (2.11) highlights one of the major advantages of quaternions for attitude propagation: The eigenaxis and eigenangle are more easily extracted from a quaternion than from a transformation matrix. The converse is also true – given an eigenaxis and amount to rotate, it is easy to find the corresponding quaternion.

MULTIPLE ROTATIONS OF REFERENCE FRAMES

Suppose we wish to further rotate $\mathbf{r}'$ by applying another unit quaternion, **p**. The resultant, purely imaginary quaternion, $\mathbf{r}''$ is found as:

$$\mathbf{r}'' = \mathbf{p}\mathbf{r}'\mathbf{p}^* = \mathbf{p}(\mathbf{q}\mathbf{r}\mathbf{q}^*)\mathbf{p}^* = (\mathbf{p}\mathbf{q})\mathbf{r}(\mathbf{p}\mathbf{q})^*$$

so that the composite rotation is represented by $\mathbf{p}$ times $\mathbf{q}$.

Quaternions can also be used to represent the orientation of one reference frame with respect to another, since $\bar{\mathbf{r}}$ can be any point within a coordinate frame. If we let $\mathbf{q}_{A\to B}$ be the quaternion representing the transformation from reference frame A to frame B, and $\mathbf{q}_{B\to C}$ be the quaternion representing the transformation from B to C, we can write:

$$\mathbf{q}_{A\to C}=\mathbf{q}_{A\to B}\mathbf{q}_{B\to C} \tag{2.13}$$

Two computational advantages of quaternions are now evident. First, it takes fewer operations to multiply two quaternions than it does to multiply rotation matrices. Second, after a series of multiplications, both the quaternion and the matrix will contain round off error. However, it is much easier to find the nearest unit quaternion (in a least squares sense) than it is to find the nearest orthonormal transformation matrix[13].

Finally, note that the quaternion describing the rotation from B to A is simply the conjugate of that from A to B:

$$\mathbf{q}_{B\to A} = \mathbf{q}^*_{A\to B}$$

since conjugating an attitude quaternion simply reverses the direction of the eigenaxis.

ATTITUDE PROPAGATION

If angular body rates are known with respect to an inertial reference frame, it is possible to propagate the attitude state over time by using quaternions. Let the current body rates be described by the vector $\omega = [\omega_x\ \omega_y\ \omega_z]$. The magnitude and direction of an attitude change over Δt can be found as:

$$\text{Eigenangle: } \phi = |\omega|\Delta t \tag{2.14}$$

$$\text{Eigenaxis: } \mathbf{e} = \begin{bmatrix} \frac{\omega_x}{|\omega|} \\ \frac{\omega_y}{|\omega|} \\ \frac{\omega_z}{|\omega|} \end{bmatrix} \tag{2.15}$$

If a vehicle changes from attitude A to attitude B during Δt, then the quaternion representing that attitude change is:

$$\mathbf{q}_{A\to B} = \begin{bmatrix} \cos\left(\frac{\phi}{2}\right) \\ \mathbf{e}_1 \sin\left(\frac{\phi}{2}\right) \\ \mathbf{e}_2 \sin\left(\frac{\phi}{2}\right) \\ \mathbf{e}_3 \sin\left(\frac{\phi}{2}\right) \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{|\omega|\Delta t}{2}\right) \\ \frac{\omega_x}{|\omega|} \sin\left(\frac{|\omega|\Delta t}{2}\right) \\ \frac{\omega_y}{|\omega|} \sin\left(\frac{|\omega|\Delta t}{2}\right) \\ \frac{\omega_z}{|\omega|} \sin\left(\frac{|\omega|\Delta t}{2}\right) \end{bmatrix} \tag{2.16}$$

This is sufficient for inertial attitude propagation. However, since we are interested in attitudes relative to a target structure, we shall use the LVLH coordinate system for reference. We must therefore account not only for the

vehicle's inertial body rates but also for the rotation of the LVLH frame within inertial space.

Reviewing figure 2-1b, we can see that the LVLH frame rotates (with respect to inertial space) about its own negative y axis once per orbit. Therefore ω_{LVLH} is $[0\ -\omega_O\ 0]^T$ and the quaternion describing the frame's rotation from time t_0 to time $t_1 = t_0 + \Delta t$ is:

$$\mathbf{q}_{LVLH0\rightarrow LVLH1} = \begin{bmatrix} cos\left(\frac{\omega_0 \Delta t}{2}\right) \\ 0 \\ sin\left(\frac{\omega_0 \Delta t}{2}\right) \\ 0 \end{bmatrix} \tag{2.17}$$

Given a current LVLH attitude at time t_0, and a set of body rates ω, our goal is to find the LVLH attitude at time t_1. The propagation strategy is:

1) Express the current attitude as a quaternion $\mathbf{q}_{LVLH0\rightarrow A}$.

2) Compute the quaternion describing the total attitude change from attitude A at time t_0 to attitude B at time t_1: $\mathbf{q}_{A\rightarrow B}$.

3) Use (2.17) to find $\mathbf{q}_{LVLH0\rightarrow LVLH1}$.

4). Apply the following series of quaternion operations[6]:

$$\mathbf{q}_{LVLH1\rightarrow LVLH0} = \mathbf{q}^{*}_{LVLH0\rightarrow LVLH1}$$

$$\mathbf{q}_{LVLH1\rightarrow A} = \mathbf{q}_{LVLH1\rightarrow LVLH0}\mathbf{q}_{LVLH0\rightarrow A}$$

$$\mathbf{q}_{LVLH1\rightarrow B} = \mathbf{q}_{LVLH1\rightarrow A}\mathbf{q}_{A\rightarrow B} \tag{2.18}$$

The new attitude quaternion, $\mathbf{q}_{LVLH1\rightarrow B}$ becomes the current quaternion in the next iteration. These quaternions can be converted to the Euler angles, ψ, θ and ϕ by the transformations described in reference 14.

Jet Plume Characteristics

The problems caused by jet plume impingement include target heating, structural contamination, induced forces and moments on the target, and direct target damage. At a given point within a flowfield, these problems may be thought of as functions of three basic parameters: heat flux (heat flow per unit area), particle density, and dynamic pressure. For a particular reaction engine, these will in turn vary primarily with the distance to the point and the angular difference between the thrust axis and the vector to the point[16].

Since the primary interest here is in trajectory generation, the details of jet exhaust flowfield modeling will not be covered. However, an overview of the plume's general shape is required to shed light on the geometry of the problem and to validate the plume cost functions used in trajectory optimization.

Since a Space Shuttle simulation will be the testbed for the A* trajectory generator, we shall use the Shuttle primary reaction control system (PRCS) plumes as examples. The general plume characteristics presented are similar to other thrusters.

Figure 2-8 is a reproduction from reference 16 of estimated plume density contours for a Shuttle PRCS motor. The dark line is along a steep drop in mass density and may thus be considered near the "edge" of the plume, but notice that small density levels persist at large angles from the thrust axis. Firings for which the thrust axis is directed at sensitive structures obviously have high contamination costs, but we must also bear in mind that

excessive firings in general can result in ambient particles drifting around the vehicle in almost any direction. Therefore, trajectories which require large amounts of maneuvering to prevent direct impingement, may result in higher contamination costs - and of course, higher fuel costs.

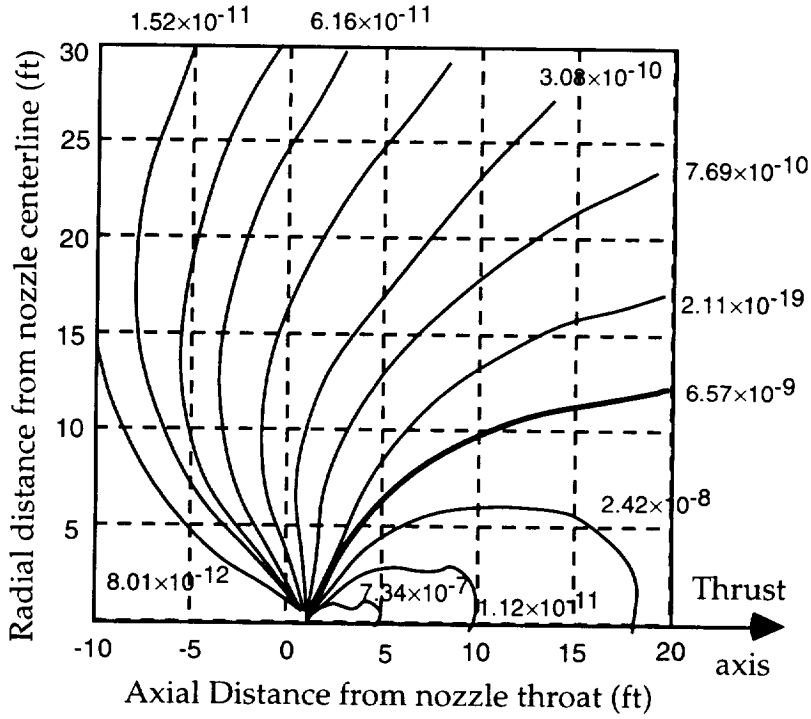


Figure 2-8. Space Shuttle RCS motor exhaust plume density contours[16].

Figure 2-9 is an iso-pressure envelope from the ICDS simulation (see chapter 4). The dark lines are 0.1 psf, and the light lines are 1.0 psf. Note the general tear-drop shape of the plume and that the plume has a circular cross-section when viewed from along the thrust axis. Also note that the base of the pressure-plume may be approximated by a 45° cone.

Experimental and computational data has revealed that - for distances greater than a few feet - exhaust plume pressure may be considered to decrease approximately as the square of the range along the thrust axis[15].

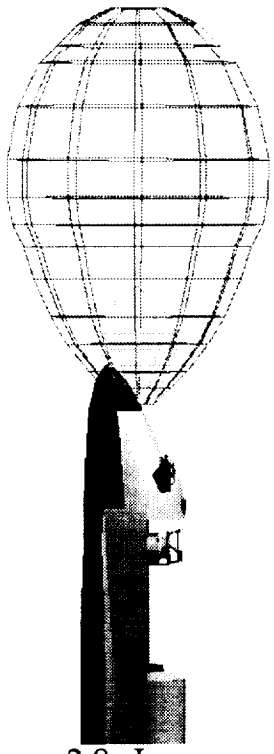


Figure 2-9. Iso-pressure envelopes of Shuttle PRCS motor.

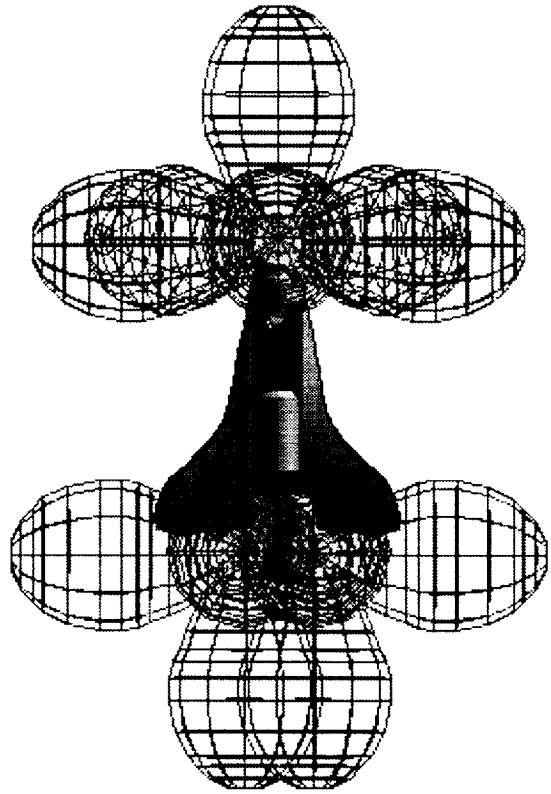


Figure 2-10. Composite Shuttle PRCS envelopes.

Figure 2-10 shows the collected plume envelopes for the Shuttle's jet configuration. It is clear that the jet exhaust pattern is by no means uniform and that the vehicle's relative attitude plays a major role in determining total plume cost. This underscores the need for six DOF trajectory planning for plume impingement reduction.

Figure 2-11 shows heat flux as a function of angle from the plume axis for a typical exhaust plume[16] . Note that all the flux is nearly zero by 10^{o} off axis, well inside the 45^{o} cone mentioned above.

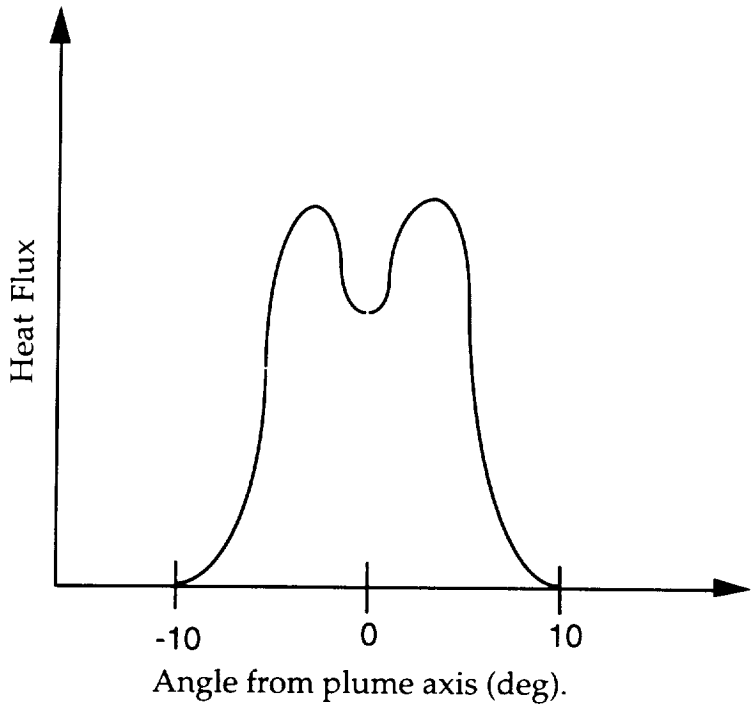


Figure 2-11. Heat flux as a function of angle off axis.

If the total cost for firing a jet at a particular structure is viewed as a weighted sum of the plume parameters at the structure, then we can make a few simple observations about the plume cost for firing that jet:

* The plume cost decreases with angle off the thrust axis and drops off dramatically outside about 45^{o} from the axis.

* Plume cost decreases approximately with the square of the range along the thrust axis.

* Iso pressure envelopes are tear-drop shaped with a circular cross section.

These characteristics are accounted for by the plume cost function presented in chapter three.

The A* Algorithm

Two general approaches to trajectory optimization have been tried for spacecraft proximity operations: gradient descent methods and node search methods. Gradient descent algorithms concentrate on the cost function, exploring its terrain, seeking the lowest point. These methods can be quite successful[4] but have problems when the cost function has local minima or is highly irregular. Reviewing the composite flowfields in figure 2-10, we can see that while it may be possible to evaluate plume costs at a particular point in state space, finding the gradient of the cost function may not be practical. Node search techniques concentrate on the trajectory space itself, dividing the space into discrete nodes and evaluating the cost at each node. Here the problem is dimensionality. To search through all possible paths through a multi-dimensional space (six dimensions in our case) in finite time, requires that the space be approximated by discrete nodes. A tradeoff exists between the density of the nodes and the computational effort required to find a solution.

Any strategy which reduces the number of nodes explored while guaranteeing optimality, will either reduce the time required to find the solution, or will allow a denser partitioning for the same amount of computational time. The A* technique discussed here is a node search technique which attempts to limit search time by directing the search along the "most promising" path.

The following sections describe the A* algorithm. Two fundamental ideas are discussed first: heuristics and decision trees. Next these ideas are

applied to an example – choosing the best path from Boston to New York. Finally, the A* algorithm is formally described, and a technique is covered for efficiently finding the cheapest element in an array, which is useful in this application. Chapter three covers the application of A* to space trajectory generation.

HEURISTICS

Heuristic knowledge is knowledge which permits selection of the "most promising" path from a set of possible paths. Such knowledge may be gained from engineering experience, problem analysis, or "common sense." Through popular usage, the noun "heuristic" has come to mean "rule of thumb" or "guiding principle." Humans use heuristics constantly. We decide whether to carry an umbrella based on the sky's appearance or which route to drive home based on our knowledge of rush hour traffic patterns.

No rule of thumb is perfect. The possibility always exists that we will get rained on, or stuck in traffic in spite of our heuristic. Here computers have a major advantage over humans. In searching for an optimal trajectory, a computer can backtrack and try another route if a high cost is encountered. The A* search is a strategy for employing heuristics and backtracking when appropriate to find the lowest cost path to the goal.

DECISION TREES AND COSTS

The decisions made in starting at a particular state and progressing toward a goal may be depicted graphically by a decision tree as in figure 2-12.

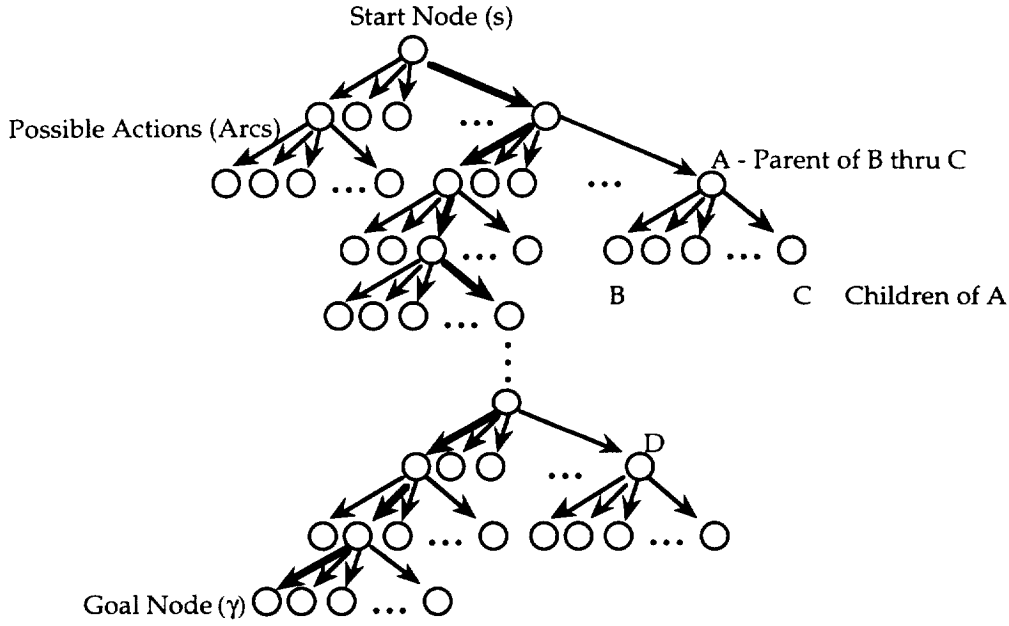


Figure 2-12. Decision tree.

Each node on the tree corresponds to a particular state, in our case an LVLH position and attitude. The lines between nodes are called arcs (even though they are depicted as straight lines) and they represent the possible actions taken at a node to arrive at a new state. Each node that has arcs emanating from it is called a *parent* node. Nodes A and D in figure 2-12 are parents. The nodes at the end of an arc are called *child* nodes – B and C are children of A in the figure. Just as with humans, it is possible for a node to be both a parent and a child. The starting point of the decision making process is the start node and the termination point is the goal node. We shall denote the start node s, the goal node γ, and any intermediate node n.

For A*, each node n, has a cost, $f(n)$, associated with it. $f(n)$ is a composite of two costs: the *generation* cost and the *heuristic* cost. These are denoted $g(n)$ and $h(n)$ respectively. The generation cost is the known cost to arrive at node n along a given trajectory. Looking at the tree, it is clear that the generation cost of a particular node, say node B in the figure, must be the generation cost of that node's parent (node A) plus the cost to go from A to B. This may be written formally as:

$$g(n) = g(p)+c(p,n) \tag{2.19}$$

where p is the parent of n and $c(p,n)$ is the cost to go from p to n.

The heuristic cost is an estimate of the cost to go from node n to the goal. The total cost at n is defined as:

$$f(n) = g(n) + h(n) \tag{2.20}$$

Notice that $f(n)$ is an estimate since it includes the heuristic estimate. An example will help shed light on the significance of these costs and their inclusion in the A* framework.

EXAMPLE: BEST PATH FROM BOSTON TO NEW YORK

Suppose we wish to drive from Boston to New York via the shortest route. We have forgotten our map, so we instruct the computer to "drive" to New York via a choice of cities shown in figure 2-13. The trees on the right of the figure show the decisions that the computer makes so that we may step through the A* algorithm. We shall define the cost function to be distance traveled. Clearly the generation cost at a city is just the distance traveled so far in arriving at that city. A method of estimating the remaining distance to

reach the goal is needed, so we choose the straight line distance from the current city to New York as a heuristic.

The computer starts at Boston and checks the distance to each of the first three cities to determine the actual distance to each. These generation costs are then added to the heuristic costs to get the total costs. Then the algorithm "drives" to the cheapest city and repeats the process. In this case, A* picks Providence because the sum of the distance from Boston to Providence, plus the straight line distance from Providence to New York was lower than the other total cost estimates.

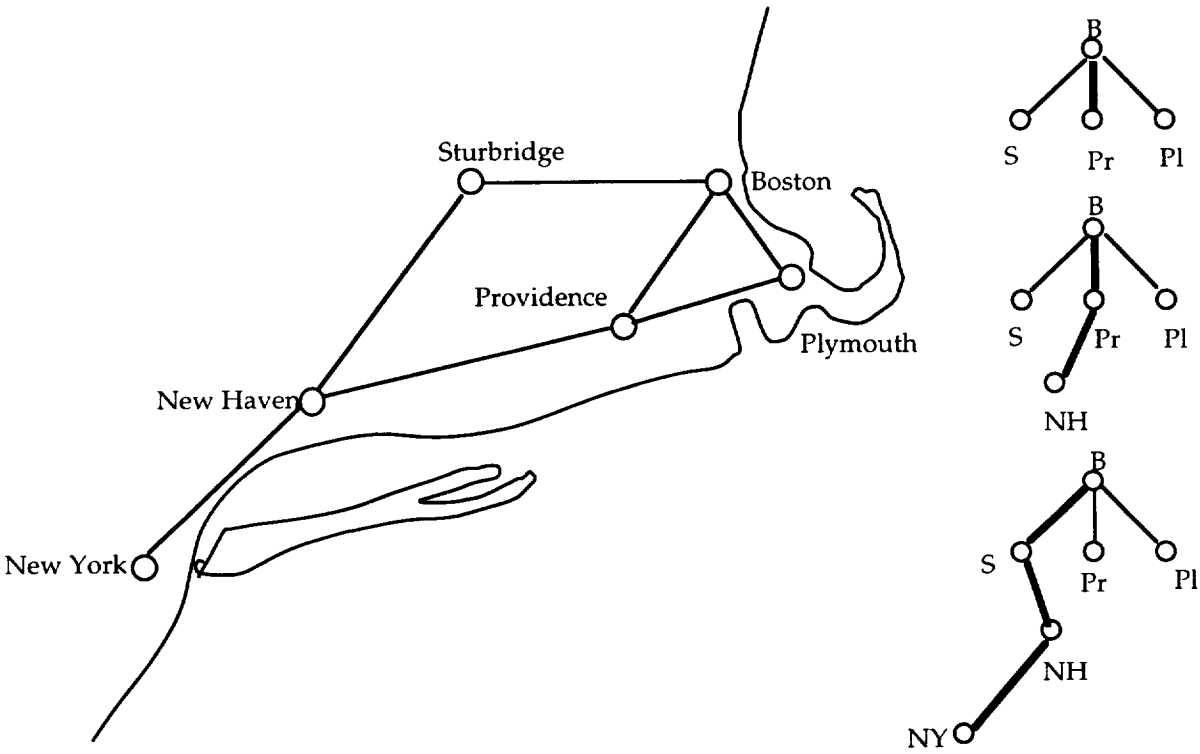


Figure 2-13. A* Example problem.

Now suppose that the road from Providence to New Haven is a crooked and hilly path so that the generation cost (actual distance traveled so far) at New Haven via Providence is higher than expected. That is

$$g(\text{New Haven}) = g(\text{Providence}) + c(\text{Providence to New Haven})$$

is larger than the heuristic prediction. This in turn may cause the total distance estimate at New Haven to rise above the total distance estimate at Sturbridge:

$$f(\text{New Haven via Providence}) > f(\text{Sturbridge}).$$

In this case, A* will backtrack to Sturbridge and continue from there.

Starting at Sturbridge, A* proceeds along the only path to New Haven. Once again, the total cost estimate at New Haven will be higher than expected since the road from Sturbridge to New Haven is not perfectly straight, nor is it perfectly aligned with New York. Now three costs are compared: the total distance estimate at New Haven via Sturbridge, the estimate at New Haven via Providence and the total distance estimate at Plymouth. In our example, the route to New Haven through Sturbridge is shorter than that through Providence so $f(\text{New Haven via Sturbridge}) < f(\text{New Haven via Providence})$.

What about Plymouth? Even though the route through Sturbridge is longer than estimated, the total cost at New Haven via Sturbridge is still lower than that at Plymouth, due to Plymouth's large heuristic component (the straight line distance from Plymouth to New York). Therefore, A* will not backtrack to Plymouth. This is reasonable since we know that the straight line distance is less than or equal to the actual distance to go. So, if the estimated distance through Plymouth is greater than the actual distance along the Sturbridge-New Haven route, then the Sturbridge-New Haven route

must be shorter. So, the solution is Boston, Sturbridge, New Haven, New York.

The heuristic estimate prevented the exploration of the route through Plymouth. The computational savings are small in this example, but for larger trees, the heuristic can prevent searching most of the branches. Had the heuristic estimate been zero, then the algorithm would have started with Plymouth and done a "breadth first" search of all possible routes. On the other hand, if the heuristic estimate had been the actual distance, then A* would have known the correct path from the start and no backtracking would have been necessary. Finally, had the heuristic overestimated the remaining distance, then A* would have chosen its route based mostly on h(n) and may have chosen the incorrect route through Providence.

A heuristic estimate that is close to the actual cost to go is called an *aggressive* estimate while one which underestimates the cost is called *conservative*. It has been shown that the more aggressive the estimate, the more efficient the search – but if the estimate exceeds the actual cost to go, then optimality is not guaranteed[11]. The converse is also true: if $h(n)$ is less than or equal to the actual cost to go, then A* will find the optimal solution. In chapter three, we shall see that it is possible to slightly overestimate the cost to go while maintaining an upper bound on the difference between the optimal solution and the A* solution.

The process of finding the children of a node and computing the cost for each child is called *node expansion*. In the example, node expansion was fairly simple because all the possible child states (cities) were known in

advance from the map. In our application, we shall have to use the translational and rotational state propagation techniques discussed above to calculate the child states and costs. Thus node expansion is computationally expensive.

A* may be viewed as an algorithm which uses heuristics to reduce the number of node expansions required to find a solution. As each node is expanded, the children of that node are placed on a list which we will call the "unexpanded list." In the tree of figure 2-12, the unexpanded nodes are at the ends of arcs with no arcs departing them: nodes B and C for example. Each node on the unexpanded list is a candidate for the next expansion. As we saw from the example, A* *always expands the node with the cheapest total cost estimate , f(n).* After nodes are expanded, they are placed on the "expanded list," from which the solution is obtained when the goal is reached. (A and D are expanded nodes.) Each node has a pointer which points to its parent, so the solution is found by starting at the goal and tracing the pointers back to the start.

The A* algorithm is[11]:

1. Put the start node s on the unexpanded list.
2. If the unexpanded list is empty, exit with failure.
3. Remove the cheapest node from the unexpanded list and make it the parent, p.
4. If p is the goal node γ, exit with the solution obtained by following the pointers from p back to s.
5. Otherwise, expand p generating all its children and attaching to them pointers back to p. For every child n of p, calculate the

total cost estimate, $f(n)$. Place all children on the unexpanded list and reorder the list so the cheapest is at the top.

6. Go to step 2.

It should be mentioned here that A* can be used in either direction in time. In reverse time the principles of the example still apply, including the impact of the heuristic estimate. In situations where the costs rise as the goal is approached (as is often the case for plume costs during docking), it is advantageous to go backward in time. That is the approach taken in chapter three.

USING A BINARY HEAP TO FIND THE CHEAPEST UNEXPANDED NODE

Since we expect to generate a large number of nodes and since we always wish to expand the cheapest, it is useful to have a means of ordering the unexpanded nodes so that the cheapest is readily available. Clearly this could be done by arranging the unexpanded list of nodes from least to most expensive – but this wastes time by providing more information than we really need. We can avoid this computational overhead by arranging the unexpanded list as a binary heap.

Figure 2-14 depicts an array in memory whose elements are arranged as a heap. In this arrangement, each element, a_i, has a lower cost than the two elements directly beneath it. Thus the lowest priced element must be a_1, the top element. Notice that the heap makes no lateral comparisons. It is entirely possible for a_4 to be less expensive than a_3, even though it is on a lower tier. The only guarantee is that the cheapest element is at the top.

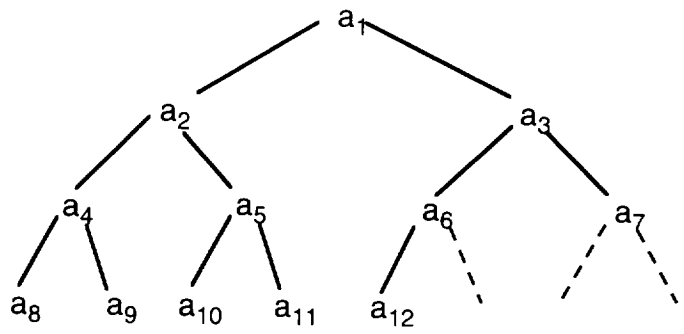


Figure 2-14. A binary Heap

If the elements of such an array point to A* nodes, then we will always have easy access to the cheapest node. When a new node is added to the heap, it is placed in an open slot at the bottom of the heap and compared to the node directly above it. If it is cheaper than its superior, the new node is "promoted," switching places with the higher node. This comparison process continues until the new node is found to be more expensive than a higher node, or it reaches the top.

From figure 2-14 we know that each layer in the heap has twice as many elements as the layer above it. This is significant because it means that the index of a particular node's superior can be found simply be *right shifting* that node's binary index. As an example, consider a_{11} in the figure. Decimal 11 is 1011 in binary. Shifting right we have 101 (base 2) = 5 (base 10), so a_5 is a_{11}'s superior. Thus, a binary heap can be reorganized quickly to find the cheapest node.

The number of nodes stored in a binary heap is:

$$N = \sum_{i=0}^{L} 2^i \tag{2.17}$$

where N is the number of nodes and L the number of layers. A heap of 13 layers can contain over 16000 nodes while requiring a maximum of 12 comparisons to rearrange.

APPLICATION OF A* TO PLUME-FUEL OPTIMAL TRAJECTORY PLANNING

CHAPTER THREE

The A* algorithm is well suited to finding an optimal path through a decision tree. In order to apply the algorithm to trajectory generation, the attitude-translation space must be discretized into a collection of nodes, each of which has associated generation and heuristic costs. This chapter details how this was accomplished for this application. An overview of the algorithm is presented first, providing a framework for the discussion. It is followed by sections on cost determination and node expansion strategies. Finally, two methods for improving convergence speed are discussed: reverse-time search, and relaxation of the optimality requirement.

A* Overview

Figure 3-1 is a block diagram of the A* algorithm as implemented here. After reading configuration information and the start and goal states, the algorithm places the start node on the unexpanded heap. Next the search loop starts, in which a parent is removed from the top of the unexpanded heap and expanded according to the rotational and translational schemes described in the following sections. At each iteration, the parent is checked to see if it is the goal node and if so, the algorithm exits with the solution.

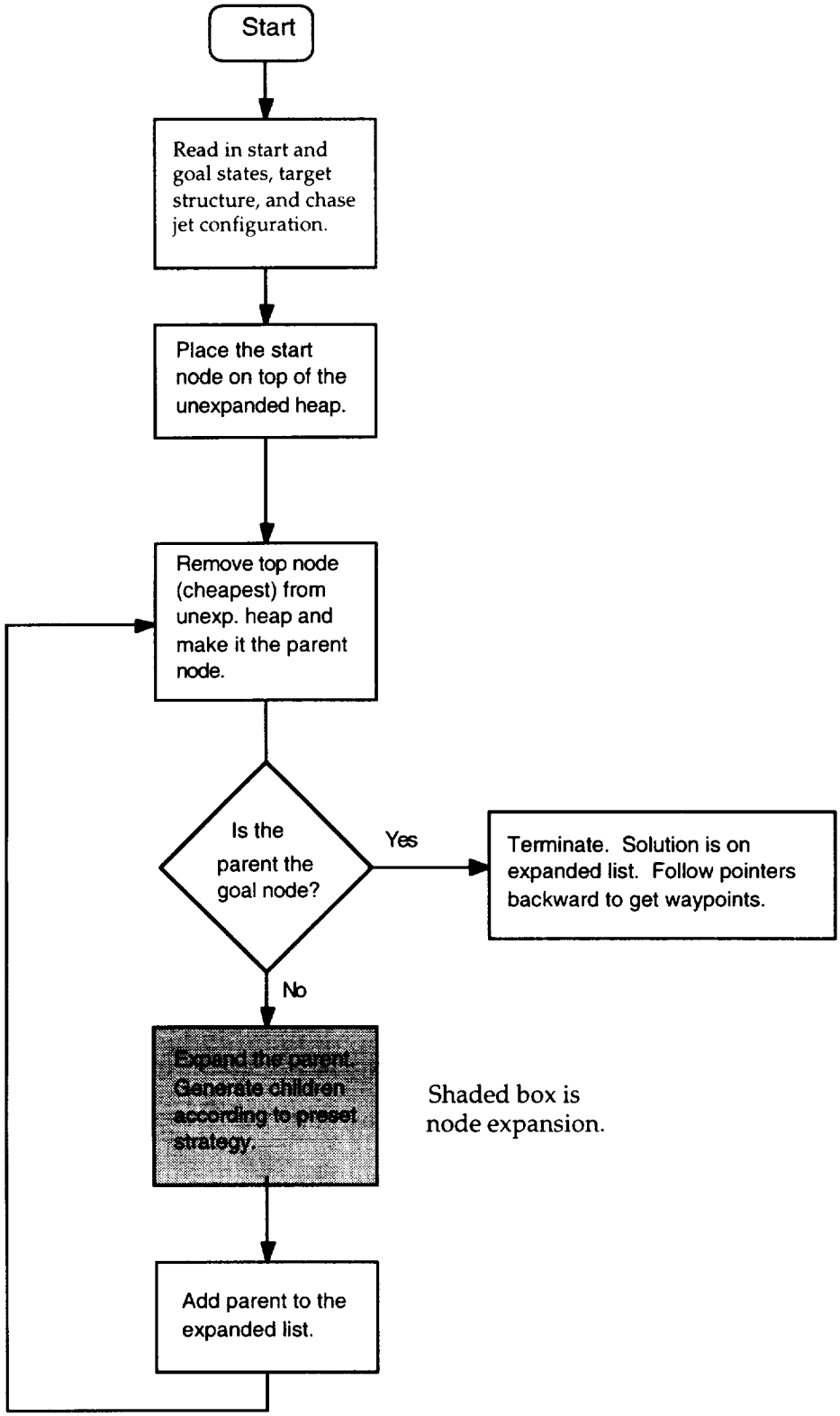


Figure 3-1. A* block diagram.

The shaded area is the node expansion process. Node expansion is a strategy for searching the state space. For this problem, it may be viewed as applying a preset sequence of velocity changes (linear and rotational) to the chase vehicle in order to provide a set of optional trajectories. Each trajectory is evaluated over the next time step to find the total cost estimate, f(n). This process consists of several steps[6,7]:

1. Generate a velocity increment according to a preset pattern.
2. Add the velocity increment to the vehicle's current velocity and propagate the state through time by Δt, using the translational and rotational methods of chapter two. This will produce a new child node. Attach a pointer to the child which points to its parent.
3. Calculate the total cost estimate of the child, f(n), for both fuel and plume costs by adding the generation and heuristic costs for each.
4. Place the child on the unexpanded heap and reorganize the heap according to cost as discussed in the section on binary heaps in chapter two.
5. Go to 1 unless there are no more children to make from this parent node.

Once the parent is fully expanded, it is placed on the expanded list.

Returning to figure 2-12, recall that the unexpanded heap consists of all the "free" child nodes – nodes at the ends of arcs that have not been expanded. It is from these nodes that A* chooses the next node to expand according to cost. All nodes that have been expanded are on the expanded list, including the

solution nodes (indicated by heavy arrows in 2-12). Once the goal is reached, the solution is removed from the expanded list by following pointers backward from the goal to the start. Each of the steps outlined above are discussed in detail in the following sections.

Cost Determination

It is useful to start the discussion of the node expansion part of A* with the manner in which costs are determined, because the ideas and philosophies behind cost assignment will also influence the manner in which the state space is discretized. The "best" trajectory is defined here as the trajectory that minimizes the weighted sum of two costs, fuel cost and plume cost. These costs must be precisely defined for A*, and their definition is driven by three factors: quality of the solution, computational efficiency, and control issues.

CONTROL ISSUES

The methods used here to determine plume and fuel costs are influenced by the fact that trajectories will be generated off line. An extremely fast trajectory generator could re-plan a new trajectory at every controller time step, but, with current hardware and search techniques, it is unlikely that any significant degree of optimality would be achieved. However, for space based use, it is desirable to find solutions in a reasonable time, so that trajectories can be generated on orbit – hence our use of A*. Off line planning requires that the trajectory be stored and followed by a control system – either an automatic feedback controller or a human pilot (or a combination). While control techniques are not the main focus here, there are some control issues that are important to trajectory generation.

The vehicles of interest are controlled by on-off thrusters. By definition, such thrusters have a fixed nozzle, so thrust modulation is not possible except by varying the on times of the jets. The jet on times are also usually restricted to fixed quanta. The Space Shuttle jets, for example, are turned on in multiples of 80 milliseconds. This in turn quantizes the forces and moments applied to the vehicle.

In a control context, this means that purely proportional control about a desired state is not possible. Instead, deadbands are defined in attitude and translation. When a deadband is reached during steady state operations, jets are commanded to fire, a rate is generated away from the deadband, and the vehicle coasts until another deadband is reached.

As discussed in the previous chapters, we shall assume that the forces and moments applied by the jets, are impulsive. Because of this assumption, we may view each jet firing as a discrete change in spacecraft velocity (often called a "delta v"). Conceptually, these delta v's are six dimensional vectors which are added to the current state to instantly change the direction of the vehicle's rotation and translation.

The attitude and translation deadbands may be visualized as a six-dimensional sphere that follows the trajectory through the state space. After the initial firings which start a vehicle along its path, the vehicle coasts for much of a typical approach trajectory. However, jet firings continue to occur whenever a deadband is violated. Small variations in initial conditions, unmodelled

disturbances, and modeling errors, make it very difficult to predict which jets will be fired due to deadbanding along a trajectory.

In planning then, we need to account for two general types of firings: deadbanding firings which occur more or less randomly throughout the trajectory, and "trajectory altering" firings which occur whenever the vehicle embarks on, or departs, a coasting trajectory. Both types of firings will incur fuel costs and plume impingement costs. At the nth node, A* will require six expressions of cost:

$g_f(n)$: fuel generation cost. $h_f(n)$: fuel heuristic estimate.

$g_{db}(n)$: deadbanding plume generation cost. $h_{db}(n)$: deadbanding heuristic.

$g_{tf}(n)$: trajectory firing plume generation cost. $h_{tf}(n)$: trajectory firing heuristic.

Fuel costs are not broken up into deadbanding and trajectory altering costs for reasons explained below. In what follows, the words "delta v" refer to the six dimensional vector which consists of linear and angular velocity changes. The vector $\Delta\mathbf{v}$ refers to the linear velocity change and $\Delta\omega$ refers to the angular rate change.

FUEL COSTS

While fuel savings are not the primary concern during close proximity operations, it is still important to penalize fuel consumption. Trajectories which are fuel optimal for a given time of flight result in a minimum of jet firings. Recall from chapter two that excessive jet firings increase the density of the particle cloud that surrounds the vehicle in space. This increases the

contamination hazard to sensitive components on both the target and the chase vehicle.

The relationship between the direction and magnitude of a commanded delta v, and the amount of fuel burned in achieving the velocity change is very complex. Reference 6 shows that the fuel cost for implementing desired rotational and translational accelerations depends on the direction of those accelerations. For example, because of the vehicle's jet layout it may be more efficient to rotate to an attitude for which translation is inexpensive, prior to executing a translational maneuver. The most accurate way to find the cost of implementing a delta v is to run the actual jet select algorithm for the chase spacecraft to determine the number of jet firings and thus the fuel cost. Unfortunately, this is computationally expensive. Since plume costs are the main thrust here, and since the expected fuel savings for trajectories inside 500 feet is minimal, fuel costs are considered to be proportional to the magnitude of the delta v's for translation and rotation. Thus the fuel cost for implementing a translational and rotational delta v is:

$$\text{fuel cost} = \tau\sqrt{\Delta v_x + \Delta v_y + \Delta v_z} + \sqrt{\Delta\omega_x + \Delta\omega_y + \Delta\omega_z} \qquad (3.1)$$

where τ is a translation scaling factor to account for the difference in rotational and translational units. It is in this sense that trajectories are described as "fuel optimal" in the following sections.

NOMINAL TRAJECTORIES

Recall from chapter one, the assumption that the time to arrive at the goal is prescribed. The fixed time of flight assumption greatly simplifies what is meant by a fuel optimal trajectory between two points in an LVLH coordinate system.

In translation, the fuel optimal trajectory between two points for a given time of flight is a two burn maneuver consisting of a jet firing to start the trajectory, and one to stop at the goal. Recall from chapter two that the $\Delta\mathbf{v}$ required to embark on a trajectory from point A to point B is:

$$\Delta\mathbf{v}_A = \mathbf{v}_{rA} - \mathbf{v}_A$$

where $\mathbf{v}_A$ is the current linear velocity at A and the required velocity, $\mathbf{v}_{rA}$ at A is:

$$\mathbf{v}_{rA} = \begin{bmatrix} \dot{x}_A \\ \dot{y}_A \\ \dot{z}_A \end{bmatrix} = \Phi_{12}^{-1}(\Delta t)\left(\begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} - \Phi_{11}(\Delta t)\begin{bmatrix} x_A \\ y_A \\ z_A \end{bmatrix}\right)$$

If there are no disturbances and no jet granularity, applying this $\Delta\mathbf{v}$ at point A will result in a perfect coasting trajectory to B. This trajectory will also be the fuel optimal trajectory (in the delta v sense described 3.1) for a given time of flight. Figure 3-2 shows such a fuel optimal trajectory for a 340 second flight from 100 feet in front of the target on V-BAR to a goal position close to the target. This trajectory requires only two major translational burns, one to start and one to stop.

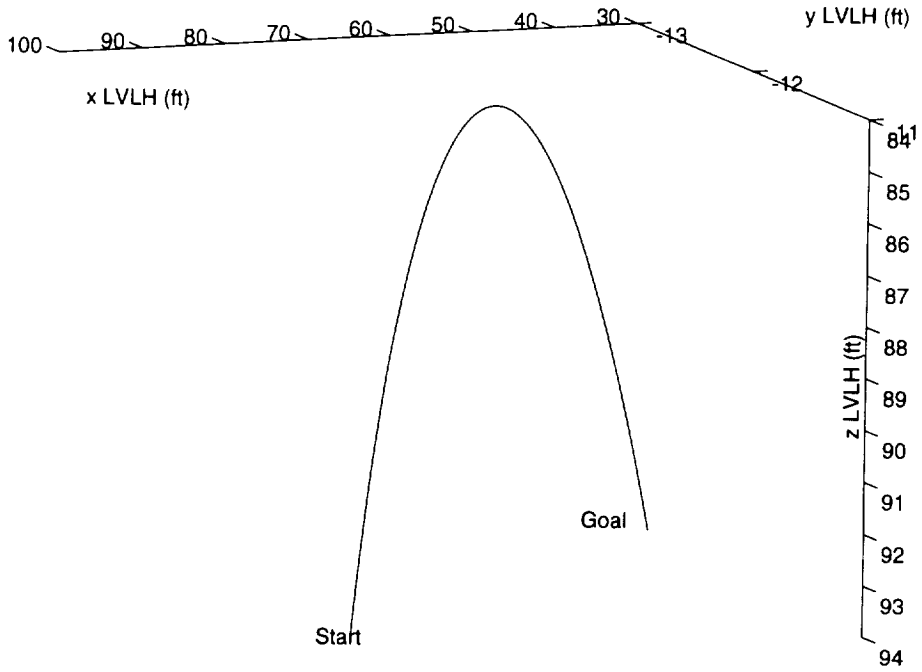


Figure 3-2. Typical fuel optimal trajectory between two points.

Of course, real jets are granular so, in executing the trajectory, the velocity change must be approximated by a sum of acceleration vectors caused by discrete firings of 80 milliseconds (in the case of the Space Shuttle). Even so, it is far preferable to plan a trajectory that takes orbital mechanics into account – both for plume impingement and fuel consumption.

In contrast, a straight line trajectory between the same two points would require that a constant force be applied to the vehicle to "fight" orbital mechanics (see chapter two). During execution, this would result in numerous discrete jet firings over the trajectory. The fuel optimal trajectory from a given node to the goal will hereafter be referred to as the *nominal* trajectory.

If Euler coupling is ignored (see chapter one), then the fuel optimal way to change from attitude A to attitude B during the time interval t_A to t_B is to fire jets

at time A to provide just enough rotation to arrive at attitude B by t_B. The magnitude of the angular velocity required is the eigenangle between A and B, ϕ, divided by the time interval (t_B - t_A). The direction of rotation is the eigenaxis, **e**. These can be computed from the attitude quaternions at A and B as described in chapter two. If B is the goal, then the nominal attitude trajectory at node n = A, is the trajectory generated by applying a $\Delta\omega$ at t_A equal to:

$$\Delta\omega_A = \omega_{rA} - \omega_A$$

where:

$$\omega_{rA} = \frac{\phi}{t_B - t_A}\mathbf{e}.$$

As discussed above, jet firings along a typical trajectory can be considered to fit into one of two categories, deadband firings and "trajectory altering" firings which cause the chase vehicle to depart from its nominal trajectory. Over long periods, the deadband firings may be considered relatively constant, so the fuel costs for deadband firings will be approximately equal over similar trajectories. Trajectory altering firings, on the other hand, are doubly expensive because once the nominal trajectory is departed, at least one more firing will be required to re-embark on a trajectory toward the goal. Thus fuel cost weighting may be viewed as a measure of our willingness to depart from a nominal trajectory in order to avoid plume impingement. Extremely low fuel costs may result in many such departures which in turn increases the contamination cloud size; a cost not directly modeled in the plume cost function described below.

To find the cost to go from a parent node to a child node, $c(n_p,n_c)$, we simply find the delta v vector required to pass from the parent state to the child state, and apply 3.1:

$$c(n_p, n_c) = \tau\sqrt{\Delta v_x + \Delta v_y + \Delta v_z} + \sqrt{\Delta\omega_x + \Delta\omega_y + \Delta\omega_z} \tag{3.2}$$

and the generation fuel cost is:

$$g_f(n) = g_f(n_p) + c(n_p, n_c) \tag{3.3}$$

The heuristic estimate of fuel cost is only slightly more complicated.

HEURISTIC ESTIMATE OF THE FUEL COST

Recall from chapter two that only conservative estimates are admissible for optimality with A*. That is, the estimated cost to go must not exceed the actual cost. However for computational efficiency, it is desirable to be as aggressive as possible in estimating the cost. Fortunately, the fuel optimal trajectory from a given node to the goal, is the nominal trajectory discussed above[6]. An acceptable heuristic estimate for fuel then, is the cost to embark on the nominal trajectory from the current state, plus the cost to stop at the goal. Thus the heuristic fuel cost can be calculated as follows:

1. Find the delta v required to change from the current velocity to that required by the nominal trajectory.
2. Propagate the state forward to the goal by the remaining time of flight.

3. Find the delta v required to change from the velocity of arrival at the goal, to the desired goal velocity. (This may not always be zero, as in a docking problem.)
4. Add the delta v's from steps 1 and 3 and apply equation (3.1) to determine the heuristic fuel cost, $h_f(n)$.

This is a highly accurate heuristic since the cost is in terms of delta v's and the state will be propagated by delta v's. In fact, once the algorithm is on a nominal trajectory to the goal, the remaining fuel costs are known exactly and no backtracking will occur unless plume costs rise or constraints are violated. As with the fuel generation cost, we assume the rate of jet firings due to deadbanding to be constant from the current node to the goal. Thus all trajectories of equal time lengths have equal deadbanding costs so there is node need to include them in our heuristic.

Our simplifying assumptions have made the fuel optimization problem simple, but the resulting computational efficiency will pay dividends when plume costs are considered.

THE PLUME IMPINGEMENT COST FUNCTION, F

Chapter two presented the characteristics of interest in jet exhaust plumes. For this application we would like to find a computationally efficient way to determine the cost for a piece of structure at a particular point within a flowfield. A simple means for representing the location of structural members and their relative sensitivity to impingement will also be required. Plume impingement

costs are vehicle specific, so the planner should take the target structural information and the chase configuration as inputs.

Our approach will be to model the structure as a collection of weighted nodes in the LVLH coordinate frame. A structural element such as a solar panel, might be represented by three to four nodes on its longitudinal axis. The nodes are weighted according to the sensitivity of the structure to plume impingement. Structures that are flimsy, sensitive to contamination, or located far from the target center of gravity – and therefore have a longer moment arm – receive higher weights.

Figure 3-3 shows a spacecraft in close proximity to a large structure. The jth jet on the vehicle is directed toward the ith structural component on the target. The assumption from chapter one is that the target has an attitude control system so that the LVLH locations of the structural nodes are known (although this code could be modified for free-floating bodies). The thin curve represents an iso-pressure envelope from the jet flowfield. Relative positions are determined by a series of coordinate transformations. Our objective is to express the relative cost for firing each jet in terms of the positions of the structural nodes within the jet's flowfield and the weights on the nodes.

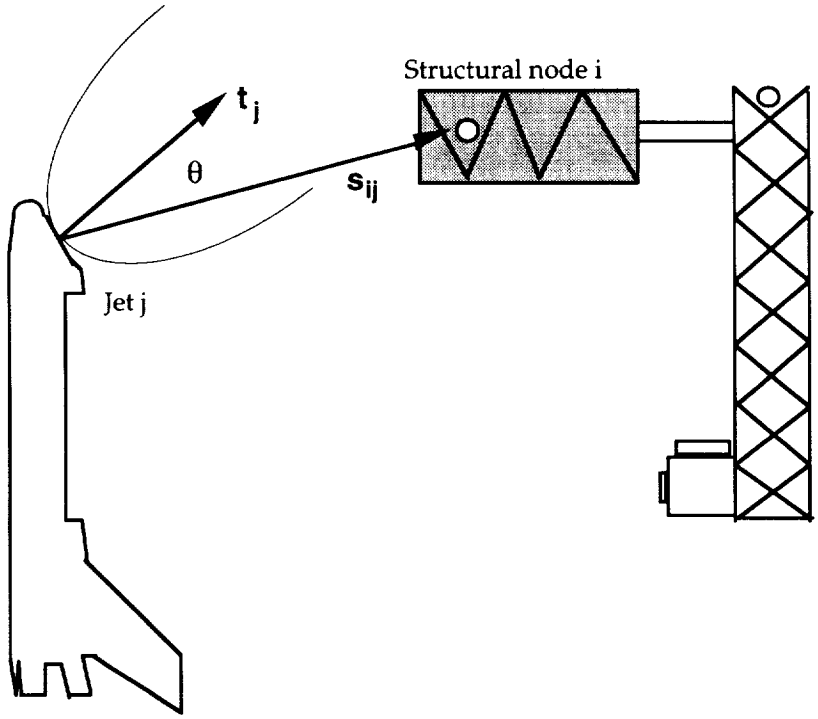


Figure 3-3. Geometric relationship between jth jet and ith structural component.

The vector $\mathbf{t}_j$ in figure 3-3 represents the magnitude and direction of a particular jet blast in LVLH coordinates. $\mathbf{s}_{ij}$ is the LVLH vector from the jth jet to the ith structure as discussed above. θ is the angle between the thrust axis and the direction of the structure. The structural weights are denoted w_i. The plume cost for firing the jth jet is here defined as:

$$p_j = \sum_{i=1}^{n} w_i F\left(\theta_i, |\mathbf{t}_j|, |\mathbf{s}_{ij}|\right) \tag{3.4}$$

where F( ) is a "plume cost function" for the ith structural node in the jth jet's flowfield, and *n* is the number of structural nodes. Notice that the cost for firing each jet is simply a weighted sum of these plume cost functions, at each node.

The function F should have properties which make the plume cost approximate the shape of the plume as presented in chapter two. The magnitude of F should drop off markedly with increasing θ since the jet plumes are directional, and F should be inversely proportional to the range squared. Since a generic spacecraft has jets with various plume strengths, **t**, it is useful to normalize the range by the magnitude of the thrust vector. We therefore define the *relative range*: $|\mathbf{s}| / |\mathbf{t}|$, and we desire F proportional to $(|\mathbf{t}| / |\mathbf{s}|)^2$.

Previous work has modeled jet plume dynamic pressure at a point as $K\cos\theta / |\mathbf{s}|^2$ (K a scaling constant)[15]. If plume costs are assumed proportional to dynamic pressure, then iso-pressure lines are also iso-cost lines and we could assign $F = K\cos\theta / |\mathbf{s}|^2$. Figure 3-4 shows two dimensional iso-pressure/cost curves for this function with a more accurate iso-pressure envelope from the ICDS simulation overlaid. This function might serve as a cost function at long ranges along the thrust axis, but would be quite conservative in regions close to the nozzle and at high angles from the thrust axis (see figure).

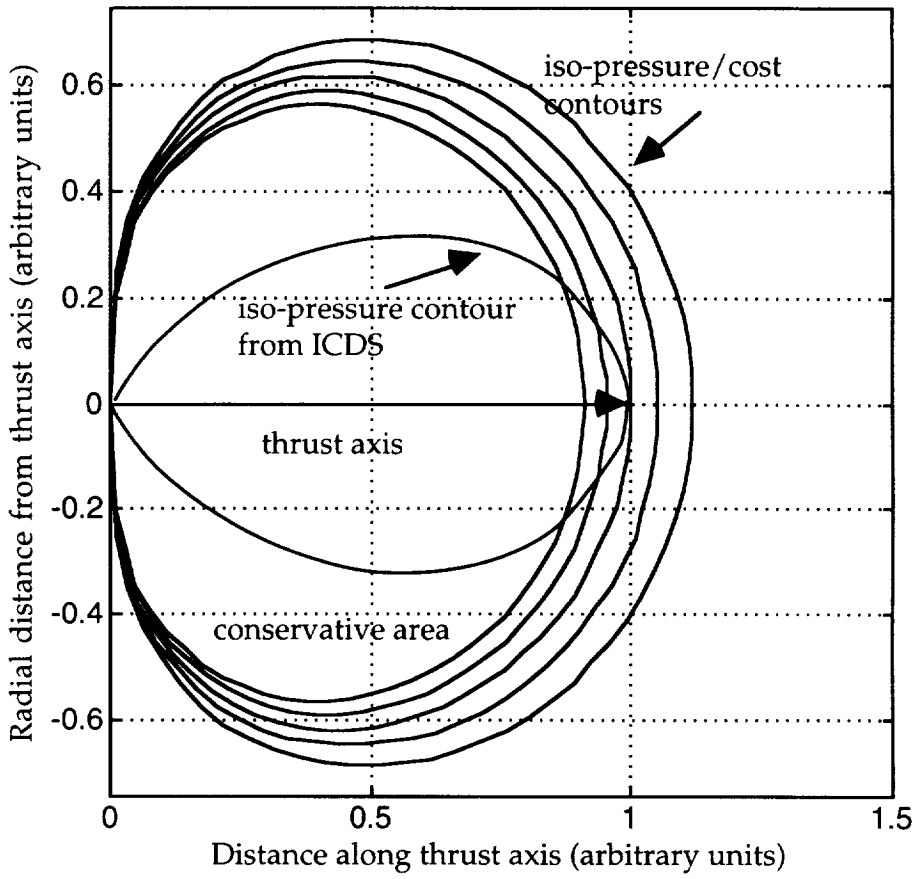


Figure 3-4. $\cos\theta / |\mathbf{s}|^2$ used as a plume cost function.

A cost function which retains the computational simplicity of the above scheme, with reduced conservatism is desirable. Consider the function:

$$F = |\mathbf{t}|^2 \cos^4\theta / |\mathbf{s}|^2 \tag{3.5}$$

The $\cos^4\theta$ factor in the numerator, causes F to decrease more rapidly with angle from the thrust axis, and contributes a teardrop shape to the function. Figure 3-5 shows a cross section of the constant plume-cost surfaces defined by 3.5.

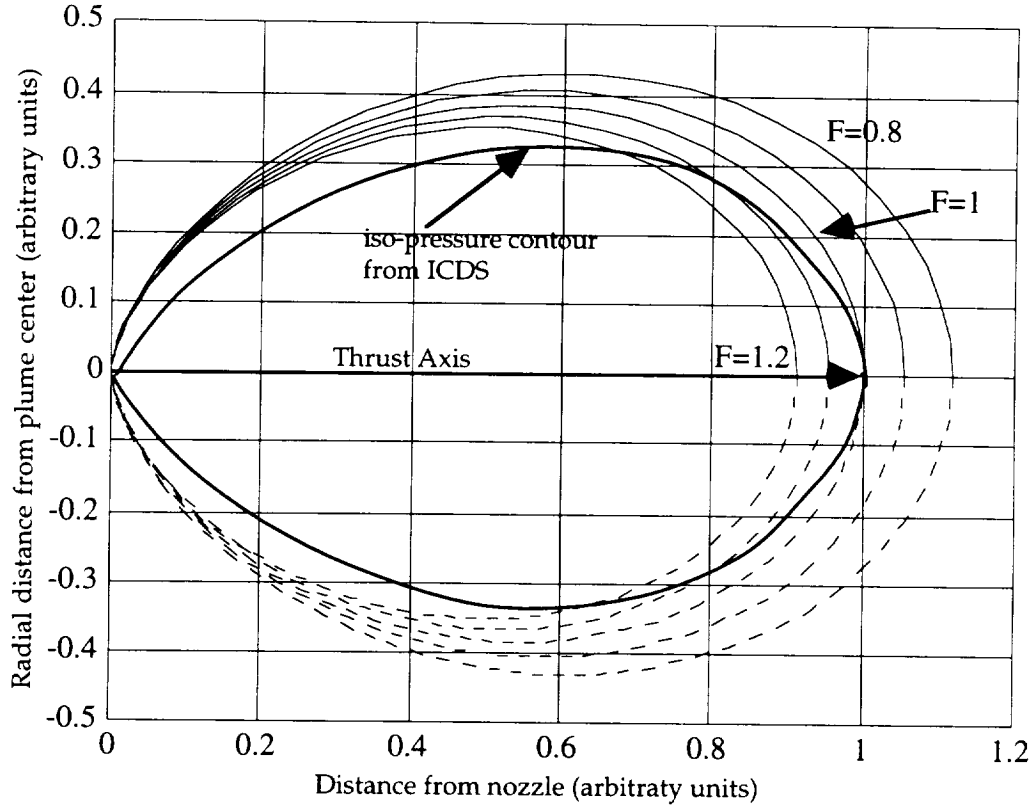


Figure 3-5. Constant plume cost lines for F about a thrust vector with iso-pressure curve.

This function is less conservative than that shown in figure 3-4; it exhibits the desired attributes from chapter two, and it can be made computationally efficient by expressing it as:

$$F = (\mathbf{t} \bullet \mathbf{s})^4 / (|\mathbf{t}|^2 (\mathbf{s} \bullet \mathbf{s})^3) \tag{3.6}$$

which contains no trigonometric functions and requires no real-time magnitude computation ($|\mathbf{t}|$ can be pre-computed).

For each jet j, there is a matrix $\mathbf{S_j}$ whose columns are each of the $\mathbf{s}_{ij}$ for the jth jet: $\left[\mathbf{s}_{1j} | \mathbf{s}_{2j} | \mathbf{s}_{3j} \ldots \mathbf{s}_{nj}\right]$. When the function F is performed on each of the

columns of **S**, we obtain a row vector, say, $\mathbf{r_j}$ whose components are expressions of the this jet's degree of plume impingement on each of the structural components. If all of the structural weights w_i are arranged in a vector **w**, the plume cost value for each jet is simply:

$$p_j = \mathbf{r} \bullet \mathbf{w} \tag{3.8}$$

Finally, the vector whose elements are the plume costs for firing each jet is defined as:

$$\mathbf{p} = [p_1 \quad p_2 \quad \cdots \quad p_m] \tag{3.9}$$

for a vehicle with *m* jets. Note that **p** varies with the state of the chase vehicle relative to the target structure, so we can expect it to vary with time as the chase follows a planned trajectory.

USING F TO ACCOUNT FOR DEADBANDING COSTS

It is tempting to try to calculate plume impingement costs during planning by simply selecting the jets required to implement a given delta v, and applying (3.8) to find the cost. However, the deadband firings discussed above often cause higher than expected plume costs when the trajectory is executed. It is quite possible for instance, for the planned trajectory to pass by a sensitive structural member with jets pointed directly at the structure. Often these jets are fired during trajectory following, even though they were not called upon in planning during the critical time when the structure was close by. Experience has shown that assigning <u>every jet</u> an impingement cost accounts for the

possibility of firing that jet during deadbanding. This cost should be related to the degree of impingement expected if a particular jet is fired.

Due to the spacecraft jet configuration, some jets may be more likely than others to be called upon at a random deadband firing. Thus, the cost for a particular jet being pointed at a structure should also be related to the likelihood of firing that jet when a deadband is encountered.

Let us assume that on the average, deadbanding firings occur at a rate of say R times per second. During a short time interval Δt, therefore, the average number of firings occurring due to deadband violations is $R\Delta t$. This means that the deadbanding cost for each time step is $R\Delta t$, times the total cost per firing.

Because the direction of a random deadband firing is assumed unknown, the total cost per firing during a particular Δt is estimated as a weighted sum of the costs for firing all the jets. If we assume the probability of the ith jet being selected during a <u>random</u> deadband firing is k_i, then the deadbanding plume cost for a given Δt is:

$$C_{db}(\Delta t) = R\Delta t \sum_{i=1}^{m} k_i p_i = R\Delta t(\mathbf{k} \bullet \mathbf{p}) \tag{3.9}$$

where p_i is the cost for firing the ith jet as before.

As the chase vehicle moves along a trajectory, the elements of $\mathbf{p}$ change as structural nodes move in and out of jet flowfields. Thus the scalar $\mathbf{k} \bullet \mathbf{p}$ can be plotted as a function (see figure 3-6). This function shall be referred to here as the "deadbanding effects" function.

Conceptually the deadbanding effects function is a measure of how many jets are pointed at pieces of important structure, weighted by the likelihood of firing each jet. We would like to minimize the height of this function over time to minimize the potential for deadbanding plume impingement.

To calculate a total deadbanding cost over an interval, the cost at each time step is calculated per (3.9) and the total summed over the number of time steps in the interval. In the limit, the deadbanding cost over a time interval t_o to t_1 is defined as:

$$C_{db} = R\int_{t_o}^{t_1}(\mathbf{k} \bullet \mathbf{p}(t))dt \tag{3.10}$$

Note that **p** is expressed as a time varying vector, and (3.10) is just the area under the deadbanding effects curve.

R (the rate of deadband firings) is assumed to be constant throughout the trajectory, so it in effect becomes a scaling factor for the deadbanding cost function (3.10). It is not necessary to find the actual value for *R* because a plume cost weight will be chosen later to balance fuel and plume considerations, and we can elect to lump *R* with the cost weight.

Figure 3-6 shows two deadbanding effects curves for a Space Shuttle docking run. The first (solid) was for a trajectory with zero plume cost weighting. The second (dashed) had a positive plume cost weighting which caused the orbiter to follow a cheaper trajectory in attitude and translation. The

characteristics of this trajectory will be discussed in chapter 4, but the figure helps to visualize how deadbanding effects can vary between trajectories.

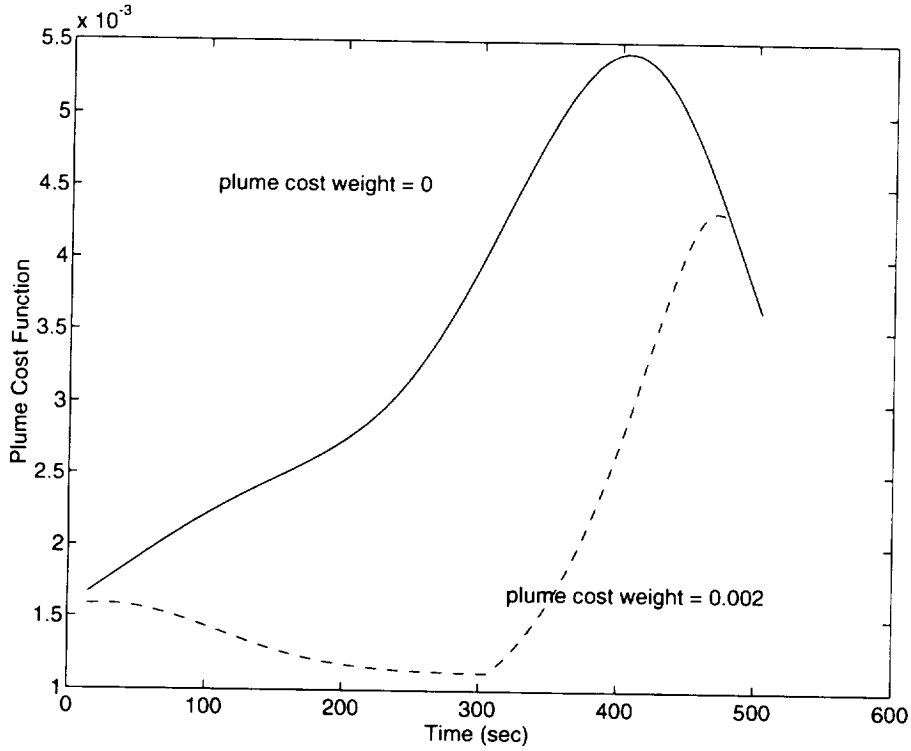


Figure 3-6. Minimizing the area beneath the deadbanding cost curve.

Note that the deadbanding effects generally increase with time on this docking run as the range between the chase and the target structure decreases. This will become important when reverse time searches are discussed.

The generation cost for deadbanding firings at node n is defined as:

$$g_{db}(n) = \sum_{1}^{s} (\mathbf{k} \bullet \mathbf{p}) \Delta t \tag{3.11}$$

where s is the number of time steps (Δt's) between the start node and node n. This approximates the integral of (3.10) up to node n. For heuristic costs, an estimate of this integral from node n to the goal is required.

HEURISTIC ESTIMATES OF DEADBANDING COSTS

The most challenging aspect of applying A* to any problem is determining a heuristic cost estimate. Our simplifying assumptions made this fairly easy for fuel costs, but for deadbanding costs, the problem is not so well defined. With fuel costs, the cheapest path to the goal was along a nominal trajectory – not so for deadbanding plume costs.

The only heuristic information we have about the deadbanding cost of (3.10) is: the deadbanding effects function does not change very much over a single time step (Δt). Figure 3-7 shows a deadbanding effects function over a typical "back away" trajectory in which the deadbanding effects decrease as the chase moves away from the target.

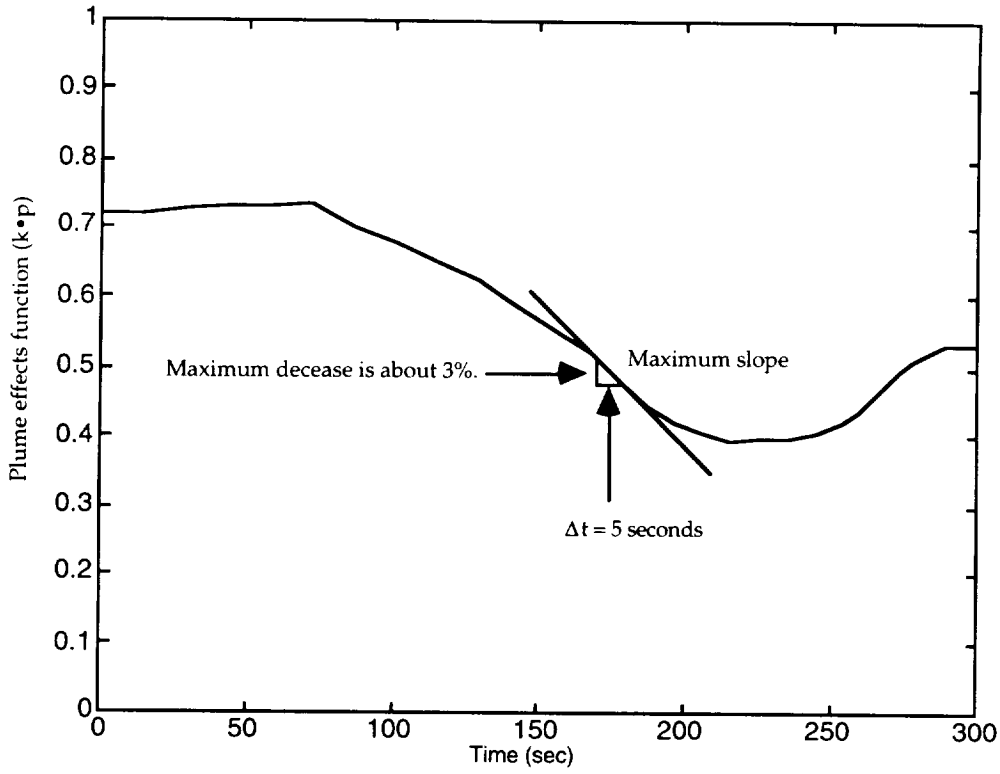


Figure 3-7. Deadbanding effects curve for a "back away" trajectory.

For this trajectory, the deadbanding effects drop-off rate does not exceed 3% over a five second time step. If a structural node is only inches away from a jet nozzle, then small rotations or translations will have a large effect on that jet's plume cost, p_i (see figure 3-5) so higher drop-off rates are possible. For the ranges of interest however, empirical testing with the Space Shuttle jet configuration has shown that the drop in the deadbanding effects function does not exceed about 3% over a five second time step. Therefore, the deadbanding effects value at each succeeding time step is at least 97% of the previous value. Let C_A be the value of the deadbanding effects function at time A so that $C_A=(\mathbf{k}\bullet\mathbf{p}(t_A))$. Then, for the Shuttle, the minimum deadbanding cost starting at time A and proceeding to the goal is:

$$C_A\Delta t + 0.97C_A\Delta t + (0.97)^2 C_A\Delta t + \ldots + (0.97)^N C_A\Delta t \tag{3.12}$$

where N is the number of time steps remaining. For a given maximum drop-off rate, r (97% in the example), the truncated geometric series of (3.12) can be summed, so that:

$$\frac{C_A\Delta t(1-r^N)}{1-r} \tag{3.13}$$

Which means that $h_d(n) = \frac{C_A\Delta t(1-r^N)}{1-r}$ is an admissible heuristic estimate to A*. This heuristic is conservative for back away trajectories, but it is even more conservative for approach trajectories. The estimate assumes that the deadbanding effects will decrease with time, but on approaches the effects increase with time. This helps motivate the reverse time search discussed below.

For this application, the elements of the vector **k** will all be assumed equal. This means that one jet is assumed as likely as another to be fired at a random deadband violation. Statistical data from multiple flights could be used to get more accurate values.

ACCOUNTING FOR TRAJECTORY ALTERING FIRINGS

The firings which change the overall path of the vehicle are more predictable, and accounting for these firings is fairly straightforward for generation costs. In what follows, these firings will be referred to as "trajectory firings." Whenever the vehicle departs or returns to a nominal trajectory, a jet

selection algorithm is run and the trajectory firing cost determined as the sum of the costs for firing each jet:

$$g_{tf}(n) = \sum_{i=1}^{m} J_i p_i = \mathbf{J} \bullet \mathbf{p} \tag{3.14}$$

where J_i is one if the ith jet is on, zero otherwise, and p_i is the plume cost for firing this jet as defined above. In other words, **J** is the vector of jets turned on or off by the jet selection routine, and **p** is the vector of plume costs.

Once the vehicle has departed its coasting trajectory, it must ultimately re-embark on a trajectory to the goal. With fuel costs, it was easy to determine the cheapest way to do this – fire jets to get back on a nominal trajectory to the goal right away. Since that strategy was the cheapest, we could use it as an admissible heuristic estimate. Unfortunately, that is not the case for plume impingement costs. Once a nominal trajectory is departed, there is no simple way to predict the best point to fire jets and start back toward the goal.

For many trajectories, the lack of a suitable heuristic to estimate trajectory firing costs is not a hindrance. In fact, the runs presented in chapter four use $h_{tf}(n)=0$. To see why this is acceptable, consider a trajectory which starts near the target and moves toward a goal far away from the structure. The first few layers of a simplified decision tree for such a trajectory are shown in figure 3-8.

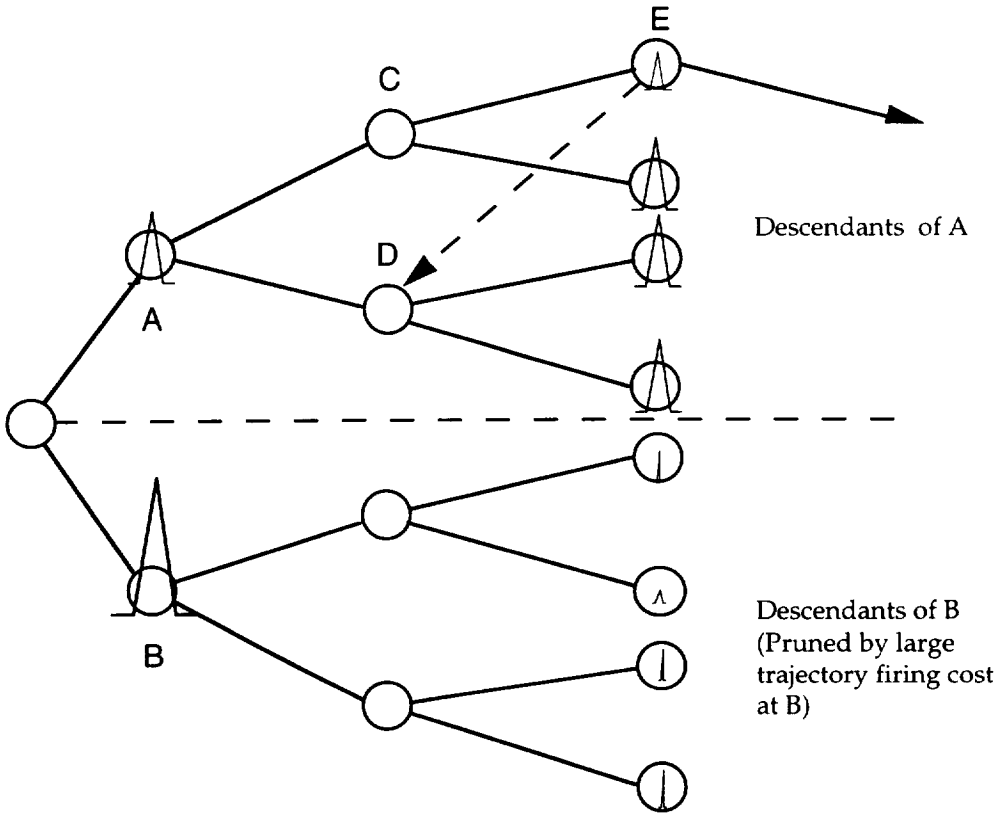


Figure 3-8. Simplified decision tree for a trajectory which starts near the target and moves away.

The circles represent nodes and the "spikes" represent trajectory firing costs encountered at each node. A node with no spike indicates that no jets are fired toward the structure at that point. Three things are important to note from the figure. First, the average size of the spikes is smaller for nodes deeper in the tree – this is expected since these nodes are farther from the structure. Second, the number of branches in the tree is small early in the search. Third, because the generation cost for each node includes the generation costs of all its ancestors, whole branches of the tree will have higher total costs due to an expensive firing early in the search. Thus, the costs at all the descendants of node B will include the large trajectory firing cost encountered at B.

Assume for the moment that node C is cheaper over all than node D. The search will proceed from A through C to E, where it will encounter an unpredicted firing cost (unpredicted because the heuristic is zero). The search will then backtrack to D (assuming all other costs are small in comparison to the firing costs at this point). Upon expanding D, the algorithm will encounter firing costs at all of D's children which will direct the search back to E to continue from there. The algorithm will only backtrack once from E, simply because the number of nodes to backtrack to is limited early in the search. If the firing costs continue to drop off as the search moves deeper, the chances are good that none of B's children will ever be expanded. This can reduce the number of iterations required for a solution, even though the heuristic estimate (for part of the cost) is zero. Thus because the highest trajectory firing costs are encountered early in the search, backtracking is limited and whole branches may be "pruned" from the tree by expensive firings.

Trajectories which start far from the structure and fly toward it can also be solved quickly by planning the trajectory backward in time (see "Reverse Time Search" below). Unfortunately, trajectories, such as fly-arounds, which have relatively constant firing and deadbanding costs, can require long search times and large memory storage capacities. Reasonable solutions may be obtained by breaking the trajectory into pieces, but a single fuel-plume optimal solution for a fly-around may not be achievable using the cost functions and node expansion strategies presented here.

Node Expansion Strategies

For many applications of A*, the state space can simply be sectioned into a grid, each vertex of which is a node. The fineness of the grid is a tradeoff between optimality of the solution and computational burden. Previous work has shown that this approach does not work well for space trajectories since for most trajectories, the spacecraft will coast on a curved arc for the majority of the time[5]. Approximating such a curve discretely, requires an extremely fine grid which is very expensive. Experience has shown that *dynamic* node expansion is preferable.

In dynamic node expansion, the locations of the child nodes are not determined until the algorithm expands the parent. As discussed above, expansion is accomplished here by applying a preset series of delta v's to the parent and propagating the state through time to find the child states. Since the nominal trajectory from a given state to the goal has benefits for both fuel consumption and contamination reduction, the expansion strategy is centered about a nominal trajectory to the goal.

NODE EXPANSION: TRANSLATIONAL VELOCITY INCREMENT

As child nodes are generated from a parent state, the nominal trajectories are determined for each child so that heuristic fuel cost estimates may be found. Since these nominal trajectories will be needed should the child become a parent, each child is tagged with the velocity vector required to embark on the nominal trajectory. Then, when a node is pulled off the unexpanded heap to become a

parent, the nominal trajectory is already known and children may be generated quickly.

The translational variation strategy used here starts with the velocity required to proceed along the nominal trajectory. This velocity is then perturbed to create a cone around the nominal trajectory. Figure 3-9 describes the pattern.

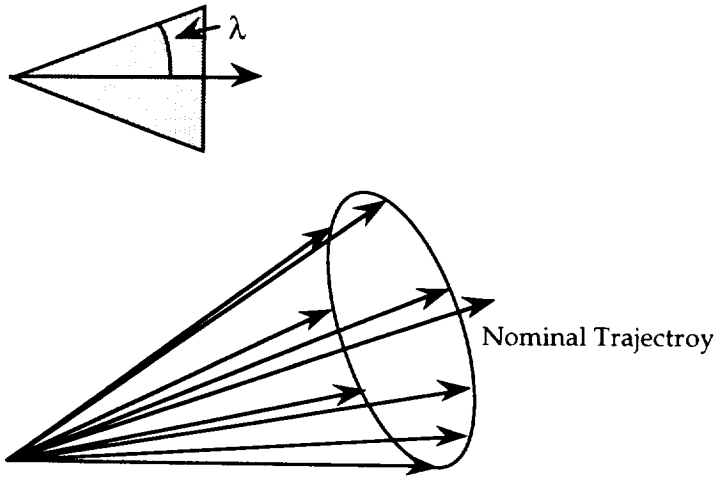


Figure 3-9. Translational velocity variation scheme.

The parameter λ in the figure expresses our degree of confidence that the optimal trajectory is close to the nominal. Smaller λ angles make it difficult to diverge from the nominal while larger angles allow drastic veers or turns in the trajectory. The number of perturbations determines the granularity of the search. $\lambda = 45^{o}$ was used to produce the results of chapter four, with eight perturbed vectors in a cone around the nominal direction for a total of nine possible directions to depart a given node. Figure 3-10 shows three node expansions to help visualize the translational strategy.

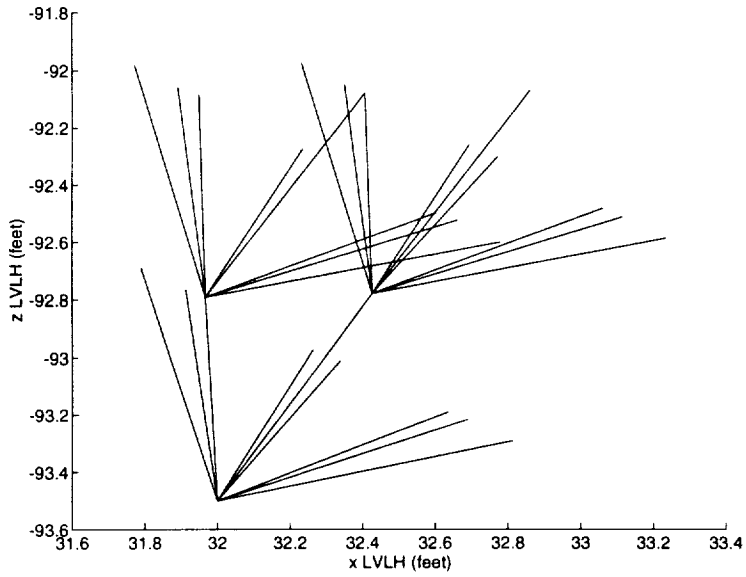


Figure 3-10. Translational node expansion strategy with $\lambda = 45^{o}$.

Figure 3-11 shows several iterations of a *breadth-first* search using these parameters. A breadth-first search occurs whenever all possible nodes are expanded at each generation. In A*, this can occur when heuristic estimates are zero (see chapter two). The figure emphasizes two points. First, this strategy covers a wide variety of possible trajectories and second, the search quickly "explodes" without heuristic direction.

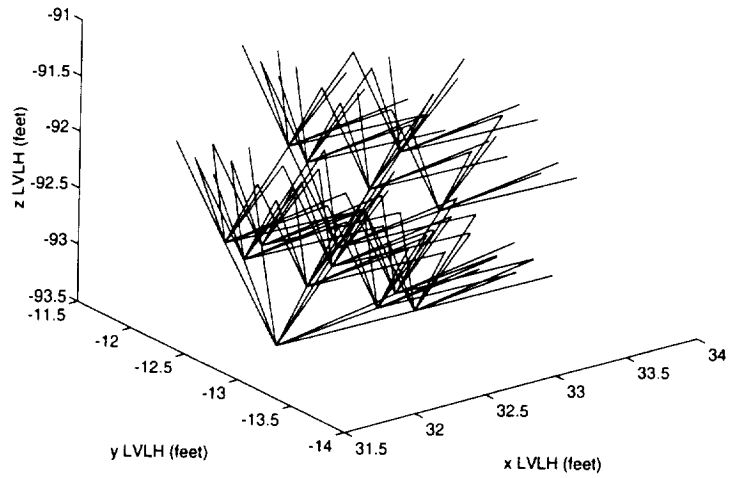


Figure 3-11. First 135 expansions of a breadth-first search.

NODE EXPANSION: ROTATIONAL VELOCITY INCREMENT

The ends of the branches in figures 3-10 and 3-11 are translational states which correspond to several nodes, each with a different rotational state. Just as with translation, the rotational search can only cover discrete nodes. The following perturbations of the nominal rotational velocities are examined at each child:

1. Allow the vehicle to coast in rotation.
2. Embark on the nominal rotational trajectory to the goal (if not already on it).
3. Execute (or stop) a ±0.2 deg/sec maneuver in roll.
4. Execute (or stop) a ±0.2 deg/sec maneuver in pitch.
5. Execute (or stop) a ±0.2 deg/sec maneuver in yaw.

Additionally, checks are made to prevent the vehicle from flying to an attitude which will require greater than a 0.2 deg/sec rate to reach the goal. The 0.2 deg/sec figure was chosen because it is a good nominal maneuver rate for the Space Shuttle so this scheme is spacecraft specific. The maneuver rates are biased by the orbital rate to allow for the rotation of the LVLH frame. The granularity of the attitude space searched depends on the time step between expansions. For a five second time step, the attitude increment is one degree (LVLH). This is excessively accurate for this application so logic was added which performs attitude expansions only every 60 seconds for a twelve degree increment.

Convergence Improvement

REVERSE TIME SEARCH

The A* algorithm is concerned only with finding the lowest cost path through a decision tree. The physics of the specific problem determine the locations of the nodes and the cost function affixes price tags but A* itself knows nothing of these. The algorithm may be used to optimize any decision process, it is up to the designer to ensure that the process relates adequately to the physical problem.

The translational and rotational state propagation equations of chapter two work equally well backward or forward in time. In other words, the equations can be used to determine the state that must have existed Δt ago given the current state and the applied control. The cost functions too are time-reversible so the trajectory can be planned equally well forward or backward in time. In backward time, node expansion becomes a process of deciding which is

the best place to come from, versus the best place to go, but A*'s decisions will be equally valid.

The sections above have alluded to the advantages of a reverse time search. Whenever the costs increase as the goal is approached, it may be advantageous to start at the goal and work backward. This prevents the computational expense of exploring a long branch of the decision tree, only to find that the path being explored has a high cost near the goal. In A* terms, this means that most of the backtracking is done near the top of the decision tree, instead of near the bottom. This is especially important when the heuristics cannot accurately predict costs ahead of time.

One particular use for a reverse time search is handling constraints, such as docking constraints. Docking constraints were imposed on some runs to show the feasibility of this approach. An example from one such run will help explain the advantages of reverse time search.

Figure 3-12 shows an A* solution trajectory for an approach to a docking port. Plume costs are not included for illustrative purposes. The translational docking constraint required that the Shuttle docking port be within a cone of the target docking port when the port-to-port distance was less than ten feet. The cone's diameter was six inches at the docking fixture, expanding to 36 inches at ten feet separation. The shaded areas in the figure violate the constraint and my thus be regarded as regions in the state space with infinite cost.

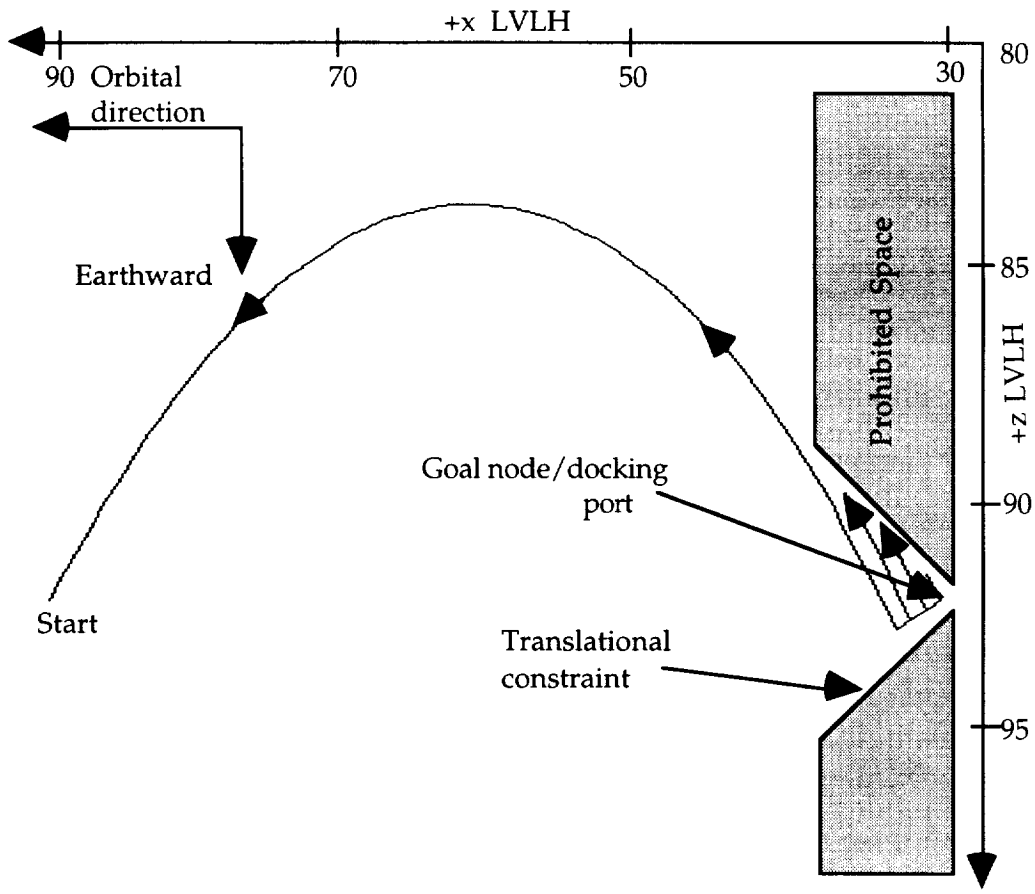


Figure 3-12. Reverse time handling of a docking constraint.

The figure is a plot of all the nodes that were expanded during the A* search. Using the city travel analogy of chapter two, the plot represents the set of cities that were tried prior to reaching a solution. The docking constraint is like a set of road blocks. When the algorithm "runs into" the docking constraint, it must backtrack and try another route.

Figure 3-13 shows the first several hundred iterations of an A* attempt to solve the same problem in forward time using the node expansion techniques described above. Convergence – if it occurs – will clearly take much longer.

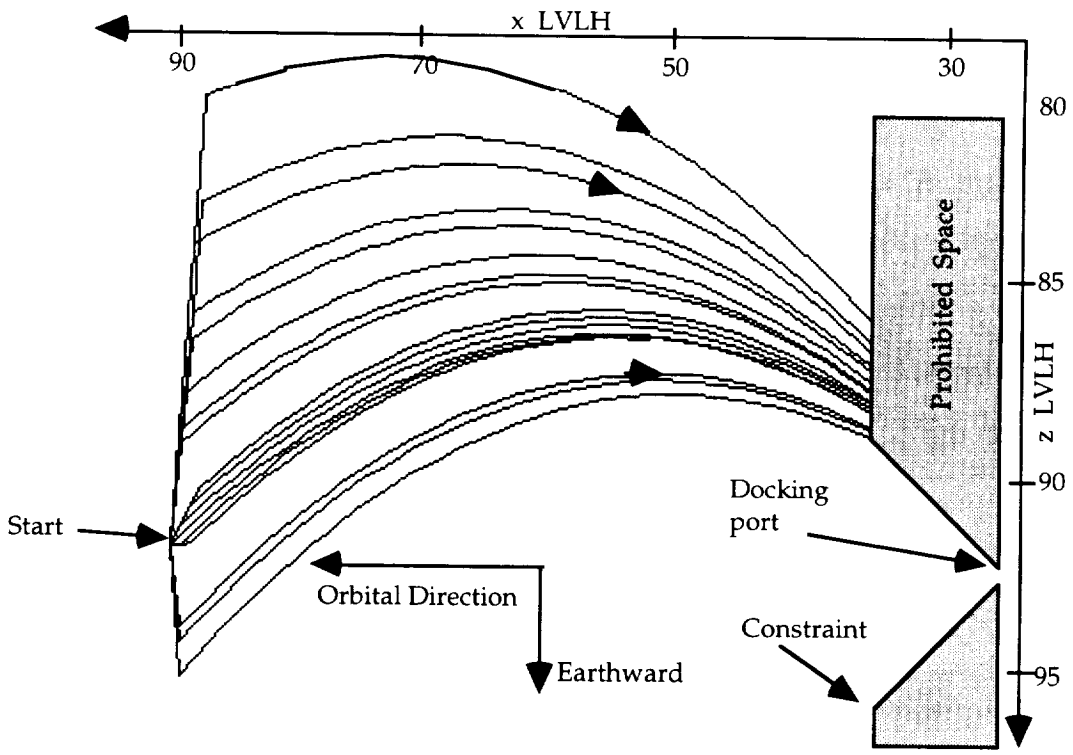


Figure 3-13. Solution attempt in forward time with docking constraint.

The reverse time search succeeds in solving this constrained problem for the same reason that we could assign a zero heuristic to costs which drop off rapidly. Nodes which violate the constraint are analogous to nodes with high trajectory firing costs, since they are expensive and they are not predicted by a heuristic estimate. Fortunately, the constraint appears early in the search because the search is conducted backward in time. Deadbanding costs also tend to rise as the structure is approached, so the deadbanding heuristic described earlier is less conservative when the trajectory is planned backward. Thus, a reverse time search can significantly reduce convergence time for both constrained and unconstrained approach trajectories.

RELAXING THE OPTIMALITY REQUIREMENT

In cases where there are many possible paths to the goal with roughly equal costs, A* can spend a large amount of time searching for the optimal path. In an application that uses dynamic node expansion, this extra searching can result in a combinatorial explosion, drastically slowing the convergence process. In such cases, a means to relax the optimality requirement while maintaining a bound on the decrease in solution quality can improve speed.

A method advanced by Pohl[18] in 1973 accomplished this by adding an additional term to the total cost estimate at each node:

$$f(n) = g(n) + h(n) + \varepsilon\left[1 - \frac{d(n)}{N}\right]h(n) \tag{3.15}$$

where $d(n)$ is the "depth" of the nth node, N is the number of generations between the start and the goal, and ε determines an upper bound on the cost increase from optimal. In this application, the depth is the number of time steps from the start to the nth node and N is the number of time steps between the start and the goal. The term $\varepsilon\left[1 - \frac{d(n)}{N}\right]$ which multiplies $h(n)$ is called a *dynamic weighting factor*. When nodes close to the start are being expanded, $d(n)/N$ is small and the dynamic weighting factor is near 1. This emphasizes the heuristic portion of the cost estimate, encouraging deeper excursions in directions that look promising. When the search nears the goal, $d(n)/N$ approaches 1 and the dynamic weighting factor fades toward zero.

If C^* is the cost for the optimal trajectory, then a search which employs (3.15) is guaranteed to produce a solution costing no more than $(1+\varepsilon)C^*$. Thus ε is an upper bound on the decrease in solution quality. Solutions obtained by using (3.15) are said to be "ε-optimal."[11,18]

RESULTS

CHAPTER FOUR

In the practical application of A*, real world problems must be expressed as decision-making processes. Two general areas determine the nature of such processes: the decisions available at each juncture, or node, and the means of determining which is the least expensive decision. For this application, these areas are rotational and translational node expansion and plume-fuel cost determination. Both of these involve simplification of the real problem, and we are interested to see how our simplifying assumptions affect the final product. This chapter describes the plant used to test this A* application and presents some results from testing. Costs predicted during planning are compared with those incurred during testing to evaluate the assumptions and strategies used, and to provide a foundation for future research.

In the future, automatic space assembly vehicles (such as those used by the Soviet space program [2]), station assembly robots and other spacecraft may use space-based trajectory planners for proximity operations. However, the Space Shuttle/Space Station combination, is the most likely near-term application, so it is used to test this algorithm. Because the Shuttle has an aircraft shape and a payload bay, the locations of thrusters are limited to certain areas, making the deadbanding effects function, of chapter three, highly irregular. Similarly, the shape of the Space Station is driven by many design factors that are in no way related to plume avoidance. These factors make the Shuttle/Station combination an interesting challenge to plume impingement reduction.

In the following sections, the pertinent aspects of the Shuttle and Station are described, along with the simulation used to generate results. The approach taken during testing was to generate a trajectory with only fuel costs considered (plume weights set to zero), and compare it to one with plume costs included. Each trajectory was executed on the ICDS simulation with a six degree of freedom autopilot to obtain plume impingement data. This was done for several start and goal states and various target configurations. Finally, some documentation on measures used to improve convergence is include.

The Space Shuttle/Space Station Combination

THE SPACE SHUTTLE

The NASA Space Transportation System (STS) commonly known as the Space Shuttle, uses two types of reaction jets for on-orbit control. The first, and most powerful, are the Primary jets which collectively make up the Primary Reaction Control System (PRCS). The second, less powerful system, is the Vernier Reaction Control System (VRCS). Figure 4-1 depicts the Space Shuttle jet configuration.

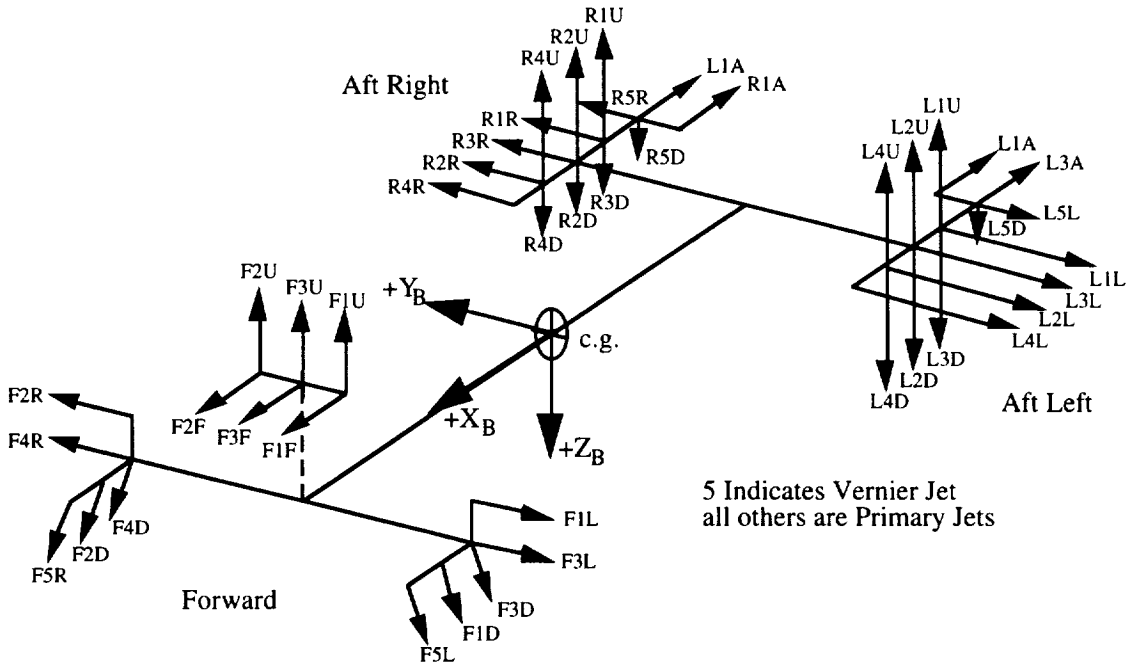


Figure 4-1. Space Shuttle jet configuration.

The jets are organized into 14 groups or clusters, numbered according to the diagram. Each jet is named by a letter/number combination which describes its location. F3D for example, is in one of the forward groups and points (generally) downward, relative to the vehicle. In the test cases described below, jets will be referred to by these names. All jets with a "5" in the name are vernier jets which are used mainly for precision attitude control. Note that no verniers point "up," so the vehicle cannot be controlled in translation using verniers only. Only the primary jets were considered in this application since they are the main contributors to impingement (see chapter five).

As the figure suggests, all the jets in a cluster have similar plume effects. This fact is used to improve computational speed during planning by treating all jets within a cluster as a single jet. If two or more jets from a single group are

fired simultaneously, their effects are assumed to be additive and the thrust vector **t**, from chapter three, is simply multiplied by the number of jets fired.

The particular jets that are called upon to execute a given delta v command depend on the vehicle mass properties and the jet selection logic. During planning, the trajectory firing costs were estimated using the lookup table for primary jets found in reference 12, because of its computational speed. This table does not take mass properties into account and therefore delivers a coarse approximation to the commanded velocity change. This in turn creates small inaccuracies in plume cost estimates. These inaccuracies were deemed acceptable since the lookup table is only used to estimate the costs for larger, trajectory altering firings, while smaller deadbanding firings were handled using the deadbanding effects function of chapter three.

Trajectory following was accomplished using a feedback controller and jet select technique developed for this project. The controller is diagrammed in figure 4-2.

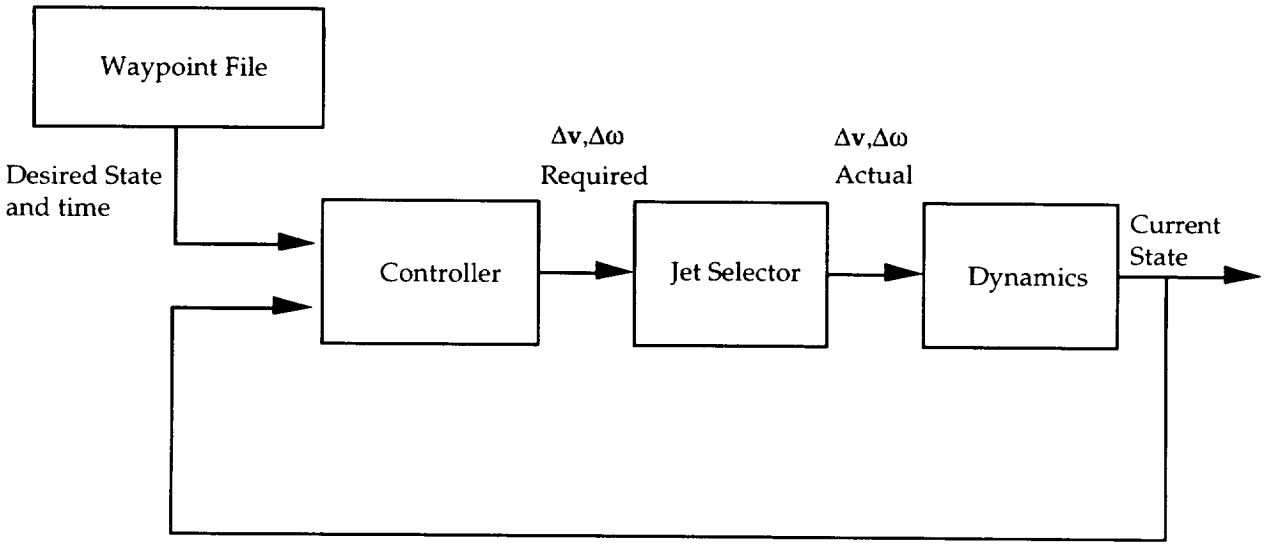


Figure 4-2. Control scheme used for trajectory following.

The controller finds the change in velocity required at each time step to arrive at the next desired state at the desired time. The controller sampling period is five seconds but the time between waypoints may be up to 60 seconds, so the controller finds the time to go to the next waypoint at each step, and uses it to compute delta v's. Required velocities are computed using equation (2.3) for translation and equations (2.14) and (2.16) for rotation. These delta v's are passed to the jet select algorithm which uses velocity deadbands to prevent continuous firings. The velocity deadbands used to generate the results below were 0.12 degrees per second and 0.02 feet per second. Over a five second sampling period, these correspond to a 0.6 degree rotational deadband and a 0.1 foot translational deadband. The same deadbands and time steps were used to test runs planned both with, and without, plume costs considered.

Perfect sensors were assumed for all the runs. Real sensors, such as NASA's developmental laser sensor, will have angular limits which may preclude some of the maneuvers generated here, but for now, these limits are not considered.

THE TARGET SPACE STATION

As of this writing, the U. S. Space Station design is in a state of flux. Current proposals vary from simple modular designs with no trusses, to complex structures adjoined to the Russian Mir Space Station. The structures used here for testing are from the original Space Station Freedom (SSF) design which was to be assembled in various stages. Two configurations were used (figure 4-3): "station configuration five" (SC-05) was used as a target for docking runs and

SC-24, a more complex structure, was used to examine other proximity operations.

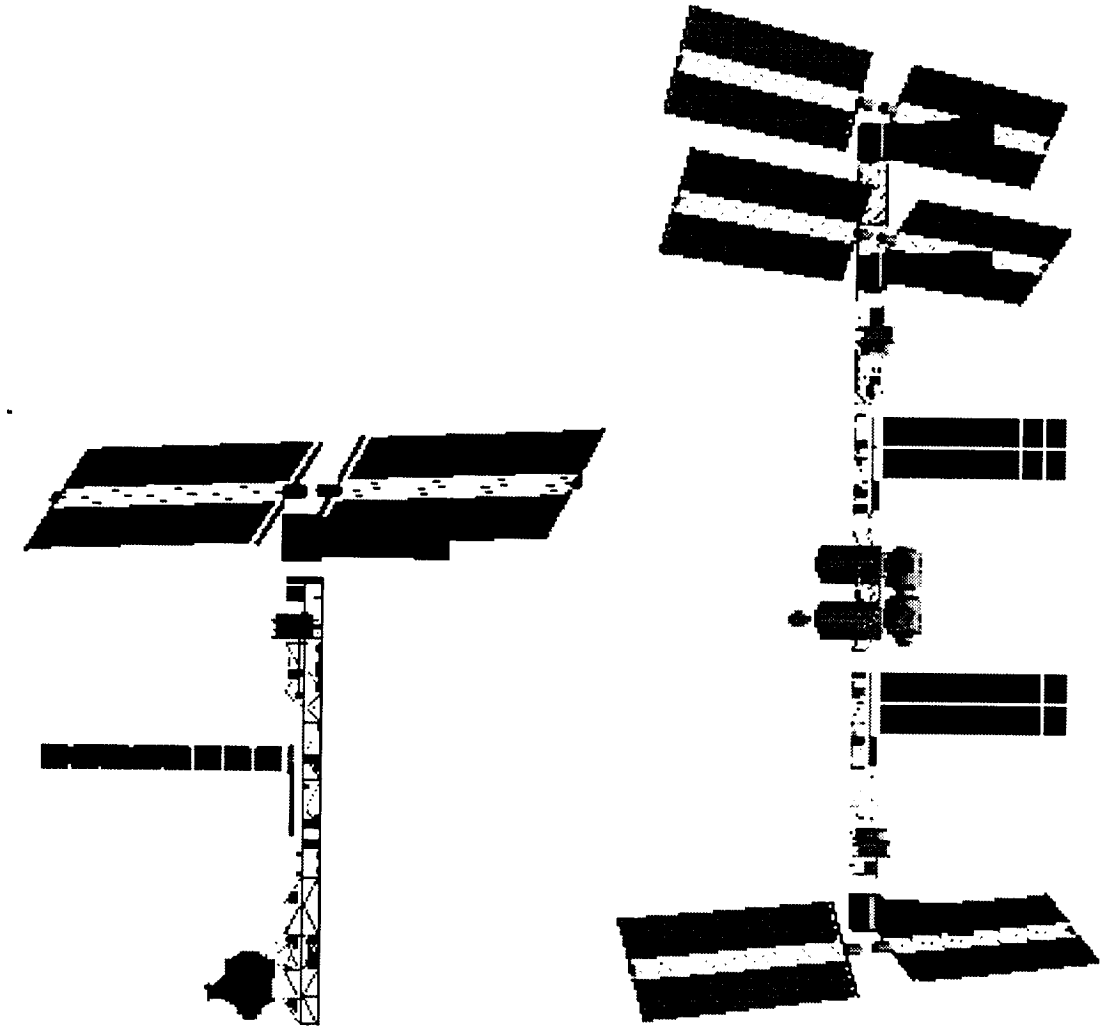


Fig. 4-3a. Side view of SC-05. Fig. 4-3b Top view of SC-24.

Figure 4-3. SC-05 and SC-24 station configurations.

Figure 4-4 shows the weighted structural nodes used to represent these configurations. The weights were assigned by a subjective evaluation of each part's sensitivity to contamination and its tendency to rotate or translate the structure when plume pressure is applied.

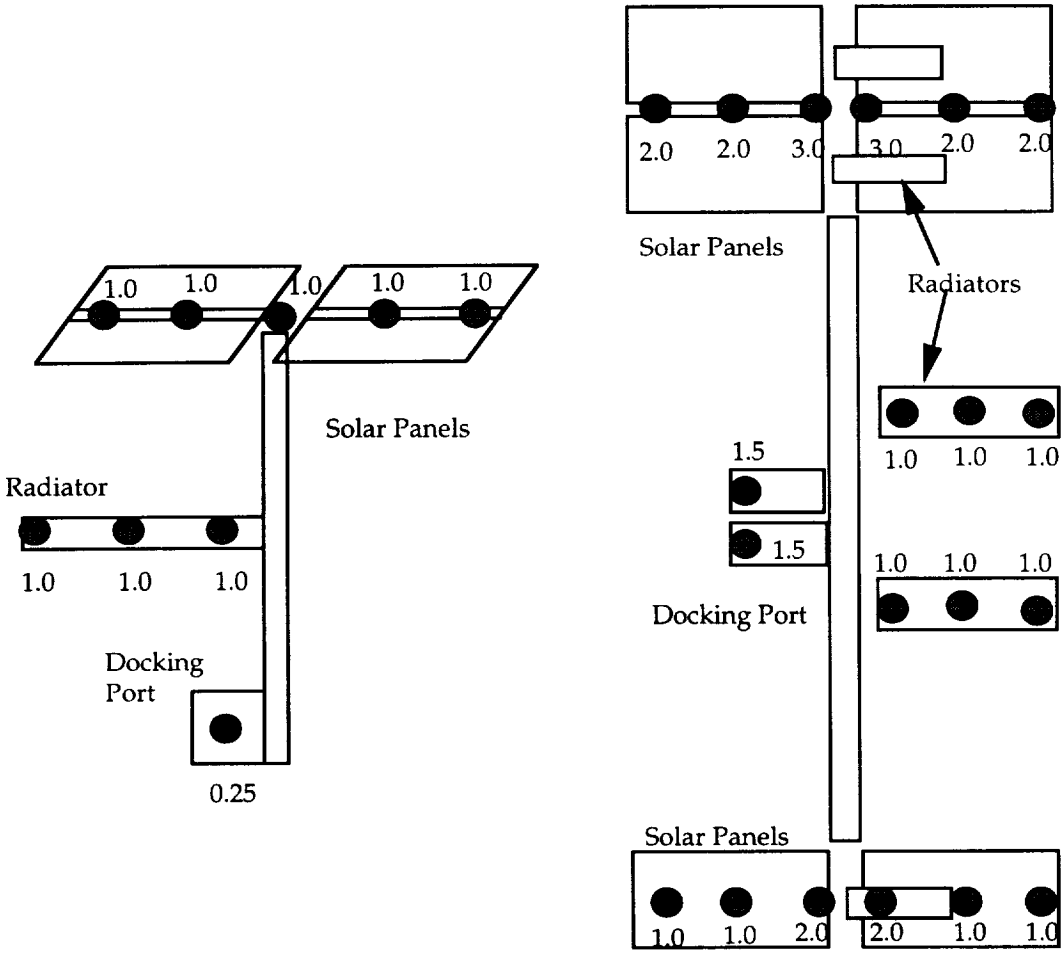


Figure 4-4. Locations and weights of structural nodes for SC-05 and SC-24.

ICDS SIMULATION

The Interactive Controls and Displays Simulation (ICDS) was used to generate the results that follow. It is a two body, six DOF, high fidelity Space Shuttle and target simulation that was developed by the Charles Stark Draper Laboratory. The graphics capability which was used to create many of the illustrations in this thesis was developed by the Lockheed Corporation under NASA contract. The simulated autopilot was modified to read a waypoint file, generate delta v commands, and implement them through jet selection logic. Several capabilities of ICDS were key to this project:

* ICDS contains a sophisticated jet plume model (developed by Norman LaFave of NASA) which generates the dynamic pressure at elemental structural locations. Each structural element is modeled as a polygon whose size and shape are generated from the graphics data. The SC-24 structure described above is described by over 44,000 polygons.

* The forces and moments from plume impingement, along with fuel consumption, jet firing histories and other pertinent data are recorded in output files.

* The graphics capability – which includes the jet plume iso-pressure "bulbs" shown in the figures – is an essential analysis tool for a six dimensional trajectory planner.

In plume impingement calculations, the component of the dynamic pressure in the direction normal to each polygon is found by taking the dot product of the pressure gradient, with a normal vector attached to the polygon. The force on each element is found as the product of the polygon's area and the normal dynamic pressure. Thus, if a structural element is parallel to the thrust axis of the jet, the plume force on the element is zero. The ICDS code was modified to allow specific structural pieces, such as sensitive solar panels, to be isolated, so that the forces applied to a particular component could be recorded.

Modeled impingement forces alone should not be used to determine plume cost results. Real structures are not infinitely thin polygons and plume

heating, turbulence, and contamination add cost, even when the surface is feathered to the jet blast. For this reason, the following analysis includes both the quantitative results of the forces applied to the structure, and subjective comments from viewing the simulation, the highlights of which are reproduced in the figures.

Trajectory Evaluation

Four cases were evaluated using ICDS. In each case, a reference trajectory was generated first, considering only fuel costs and any constraints such as docking constraints. Positive plume cost weights were then used to plan a plume efficient trajectory, and the results compared with the reference.

The format for presenting the data will be consistent for all the cases. First, the plume weighted trajectory (in attitude and translation) is presented and compared with the reference trajectory. Second, the predicted costs are analyzed to determine why A* chose a particular route. Third, both trajectories are "flown" on the ICDS simulation and the applied forces are compared with the predicted costs. Finally, each trajectory is shown as a sequence of pictures for visualization and further analysis. It is recommended that these figures be viewed prior to reading each section to help visualize the trajectories generated. The text will refer to these pictures during trajectory descriptions.

CASE 1: VALIDATION TRAJECTORY

This first test does not necessarily conform to any common or expected Shuttle proximity maneuver. Rather, it is designed to validate the planner by

presenting it with a highly constrained scenario. The trajectory starts with the orbiter 200 feet in front (in the orbital direction) of the SC-24 space station. The initial attitude has the wings parallel to the tangent plane, with the nose pointing in the $-y_{LVLH}$ direction. The goal position is close to the station with a large solar panel nearby (see figure 4-12a at the end of this section). The goal attitude is nose up with wings parallel to the y-z plane as shown in the figure. For validation purposes, the structure is treated as consisting only of the solar panel and radiator shown in figure 4-5.

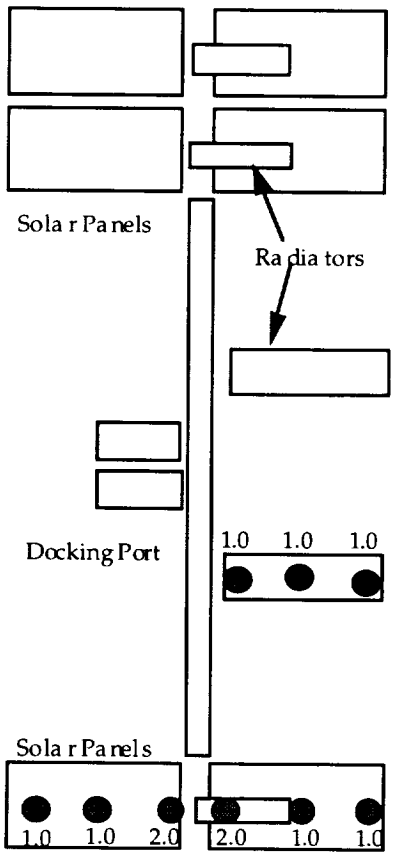


Figure 4-5. Structural weightings for validation run.

At the goal position and attitude, the orbiter has jets both above and below the weighted solar panel. The challenge is to arrive at the goal without point-blank firings at the solar panel or radiator.

C-2.

Figure 4-6a shows the reference trajectory and the plume weighted solution, for translation in the LVLH coordinate system. The units for all axes are feet. Notice that the reference trajectory stays close to the tangent plane (z_{LVLH} = 0) while the weighted path starts by rising up out of the tangent plane and then thrusting onto a coasting trajectory. The weighted trajectory also curves away from the solar panel slightly.

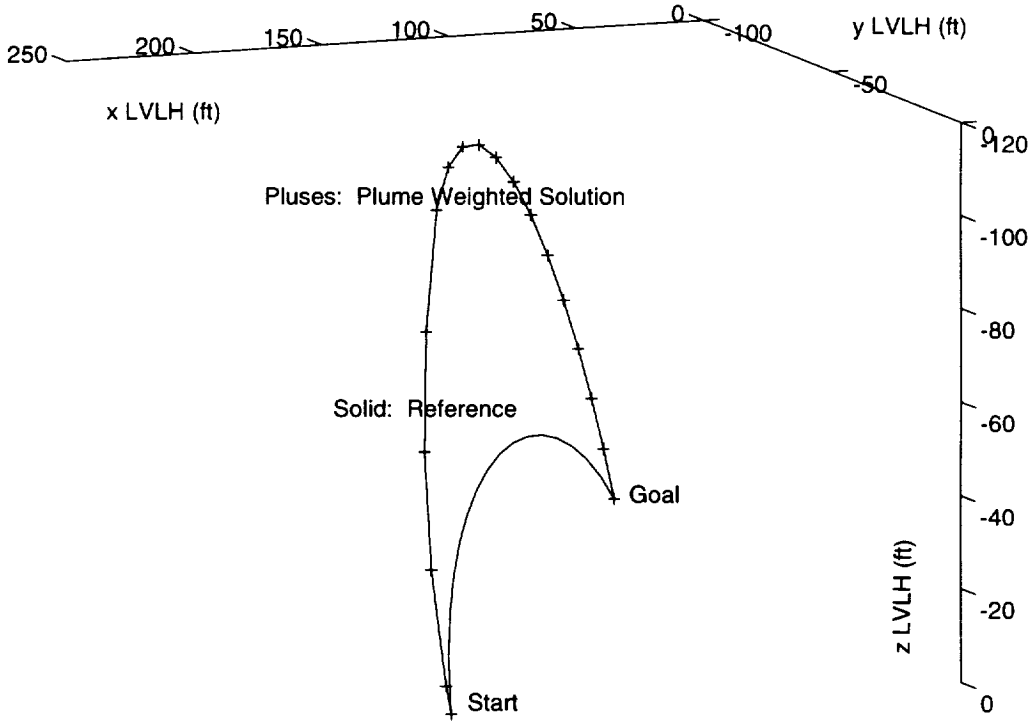


Figure 4-6a. Reference solution and plume weighted solution.

As a matter of interest, figure 4-6b is included to show the A* decision making process. All the "branches" shown correspond to expanded nodes. The plus symbols mark the solution trajectory. The nature of the backward time search is seen in the way that the decision tree branches out from the goal, to the start.

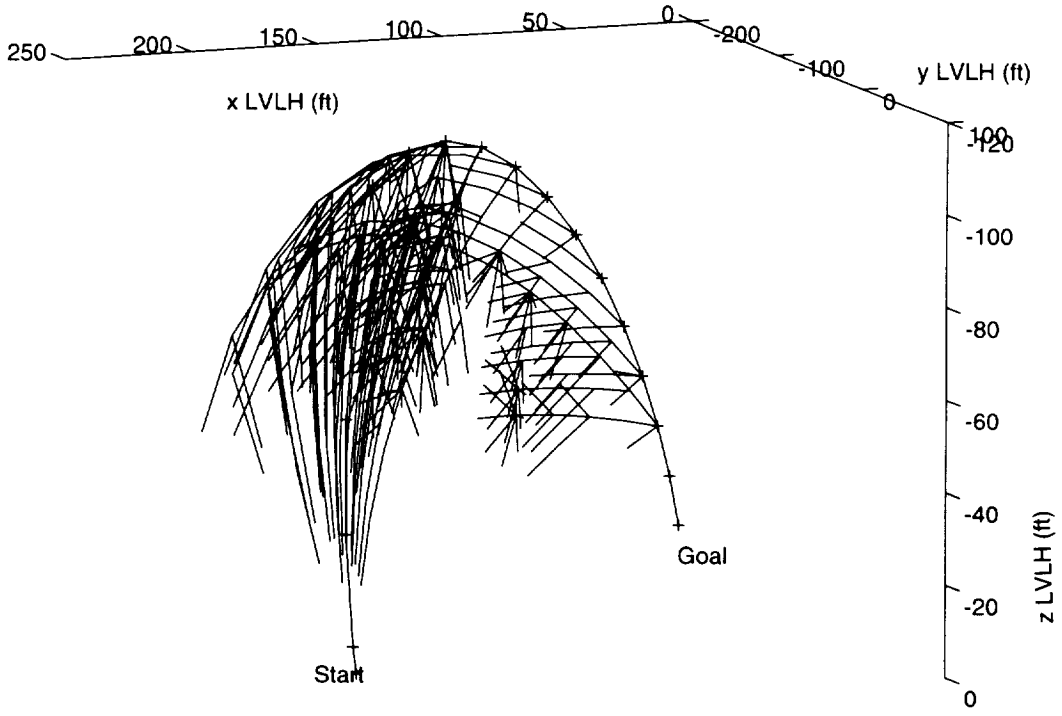


Figure 4-6b. A* branching during planning.

Figure 4-7 compares the reference and weighted trajectories in attitude. The abrupt "jumps" in attitude at the goal are due to the fact that the Euler angles are not unique at the goal attitude. Figures 4-12b and c help to visualize the attitude trajectories. Note that the weighted trajectory attempts to pass the solar panel without pointing jets directly toward it.

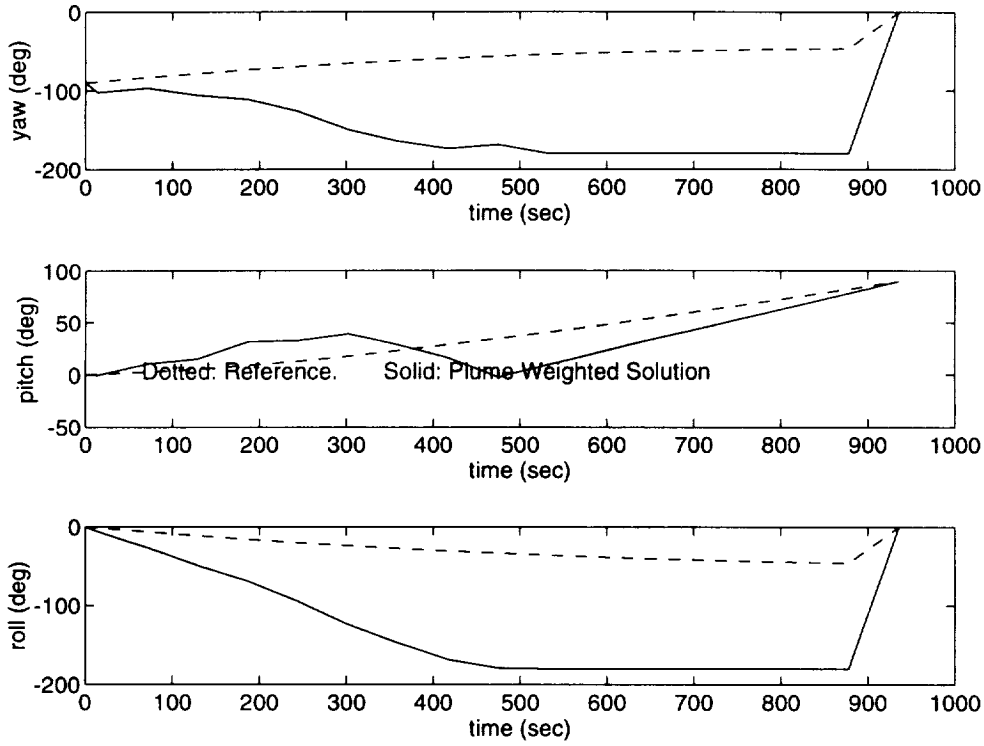


Figure 4-7. Attitude Trajectories.

Notice that the weighted trajectory arrives at the goal attitude in yaw and roll by about 540 seconds. This aligns the Shuttle fuselage as shown in figure 4-12c, so that the side-facing jets are positioned above and below the solar panel. As the orbiter translates toward the goal, a maneuver in pitch brings the tail into position.

The reasons for the above departures from the nominal trajectory become clear when the cost functions of figure 4-8 are considered. The final planned

trajectory represents an attempt to trade off the three principal costs of chapter three: deadbanding effects, trajectory firing costs and fuel consumption (as measured by delta v magnitudes). Deadbanding costs are influenced mostly by the actual vehicle state in the neighborhood of the structure – in other words, position and attitude determine how many jets are pointing toward the solar array or radiator. The cost for altering the trajectory, on the other hand, is influenced by the direction of the change in state. In this case, the best example of trajectory altering occurs at the end of the trajectory where the vehicle is required to stop. Notice from figure 4-8 that the planner has found a way to arrive at the goal which significantly reduces the cost of firing jets to stop. Recall from chapter three that this arrival direction is actually found first because the planning is done backward in time.

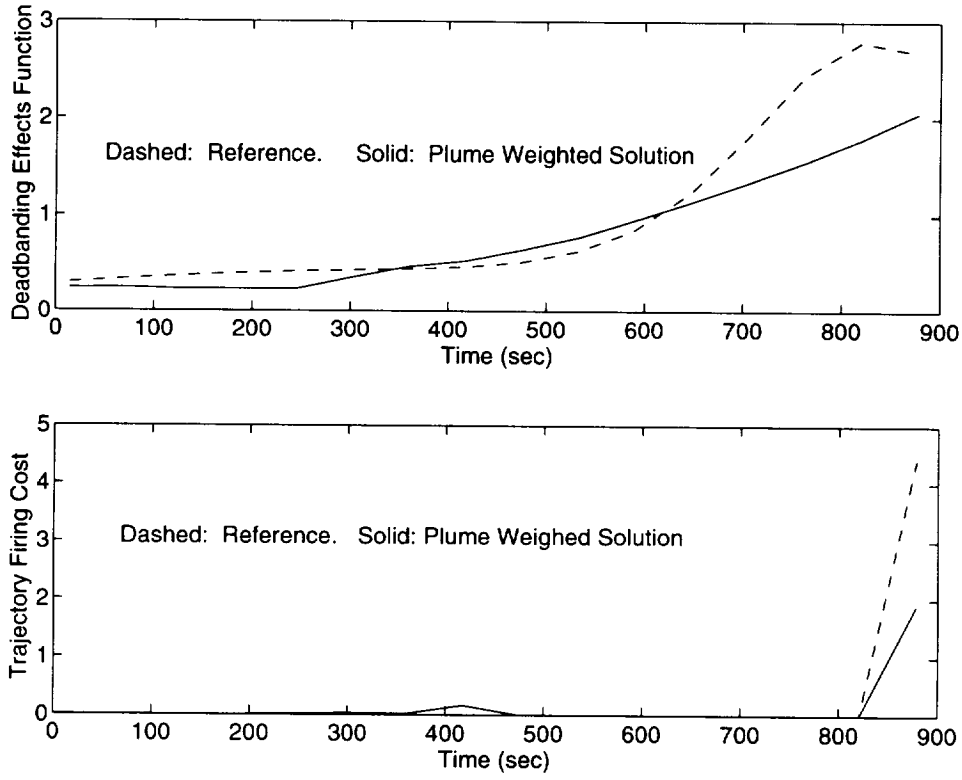


Figure 4-8. Predicted plume costs.

Referring again to figure 4-12 at the end of this section, it interesting to contrast the trajectories planned with and without plume effects considered. Figure 4-12b shows direct plume impingement caused by deadbanding firings as the orbiter passed the solar panel. Figure 4-12c shows how these are avoided in the weighted solution. The Shuttle maintains an attitude and flight path which causes the jets at the nose of the vehicle to stay above the panel while those at the tail remain below it. This attitude is also "inexpensive" for deadbanding firings toward the radiator. As the nose of the shuttle approached the goal, the attitude is changed so that the tail "swings" down and into position.

The resulting forces applied to the solar panel are shown in figure 4-9. Each spike represents the total force produced when the jets respond to a delta v command. The large spikes in the reference (dotted) trajectory from about 800 to 900 seconds are from the deadbanding firings discussed above. The 15 lb spike at the end of the trajectory corresponds to the predicted firing cost for stopping at the goal. Note that it is smaller than the deadbanding spikes. Even though the magnitude of the trajectory altering firings are generally larger than deadbanding firings, the direction of these firings may or may not cause large impingement. Note also that the predicted spike at the goal is absent in the weighted solution. This can be traced to a particular jet selection which was made when the vehicle arrived at the goal. As the final rotation was stopped, one or both of the forward, down-facing jets (F3D or F4D) were fired. F3D pointed directly at the solar panel while F4D pointed away. The choice depended on small rates in other axes. If F3D was fired during execution, a spike appeared, if not, the final firing produced virtually no cost. It is interesting to note that changes in the initial conditions, 200 feet away from the goal,

influenced whether or not this jet was fired upon stopping. This further underscores the unpredictable nature of plume impingement.

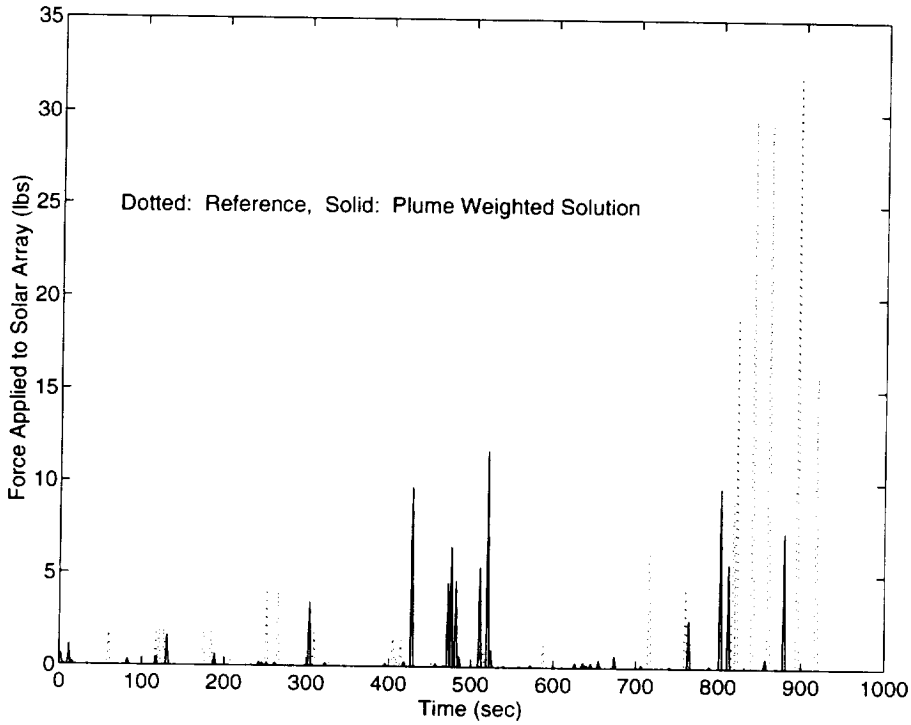


Figure 4-9. Forces seen at the solar array due to jet firings.

Figure 4-10 shows the cumulative force applied to the array for both trajectories. The total force applied is reduced in the weighted version by over 100%, due mainly to the costly deadbanding firings at the end of the reference trajectory. However, in cases where forward down jets on the left side (e.g. F3D) were used to stop, the difference was reduced to 40% to 60%. The region on the cumulative force plot where the plume weighted trajectory rises above the reference corresponds roughly to the predicted deadbanding curves which show the same relationship.

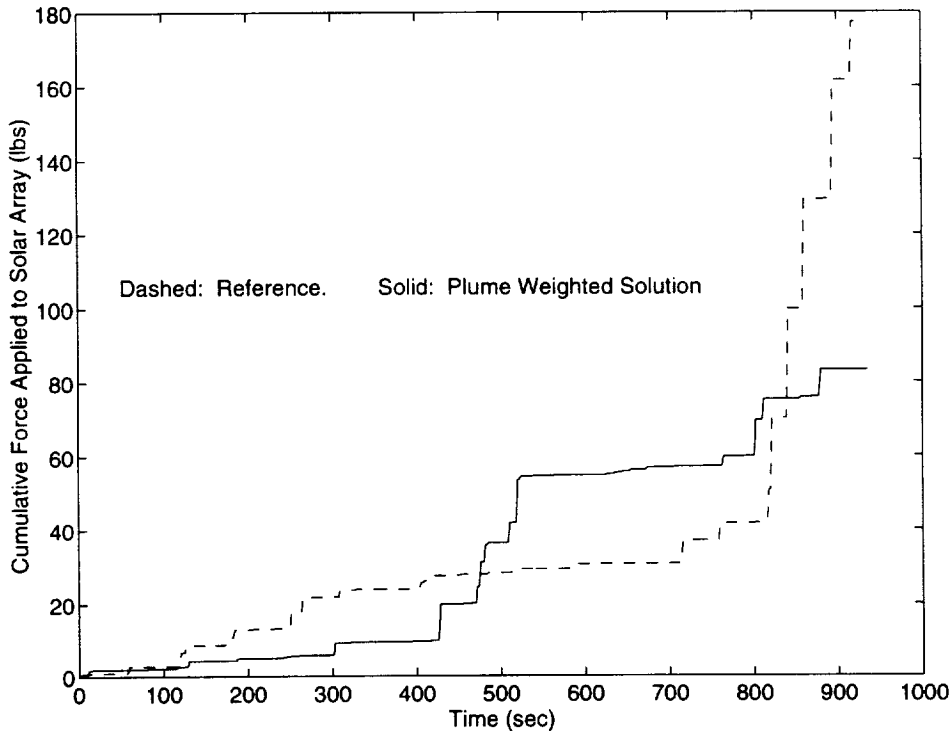


Figure 4-10. Cumulative force.

Figure 4-11 shows the main tradeoff. Cumulative 80 ms jet firings are depicted for each trajectory. These may be regarded as approximately proportional to fuel consumption, and they also relate to the density of the "contamination cloud" alluded to in previous chapters. The price for reduced force on the panel was an increase of about 25% in total jet firings. It should be noted here however that the number of firings is a function of the control technique as well as the trajectory. In this, and all cases presented here, jet select deadbands are constant at the values described above. This causes a relatively large number of deadband firings at all ranges for both trajectories – which in turn tends to belittle the fuel costs incurred when the trajectory is altered (see chapter 5, *Controller Improvements*).

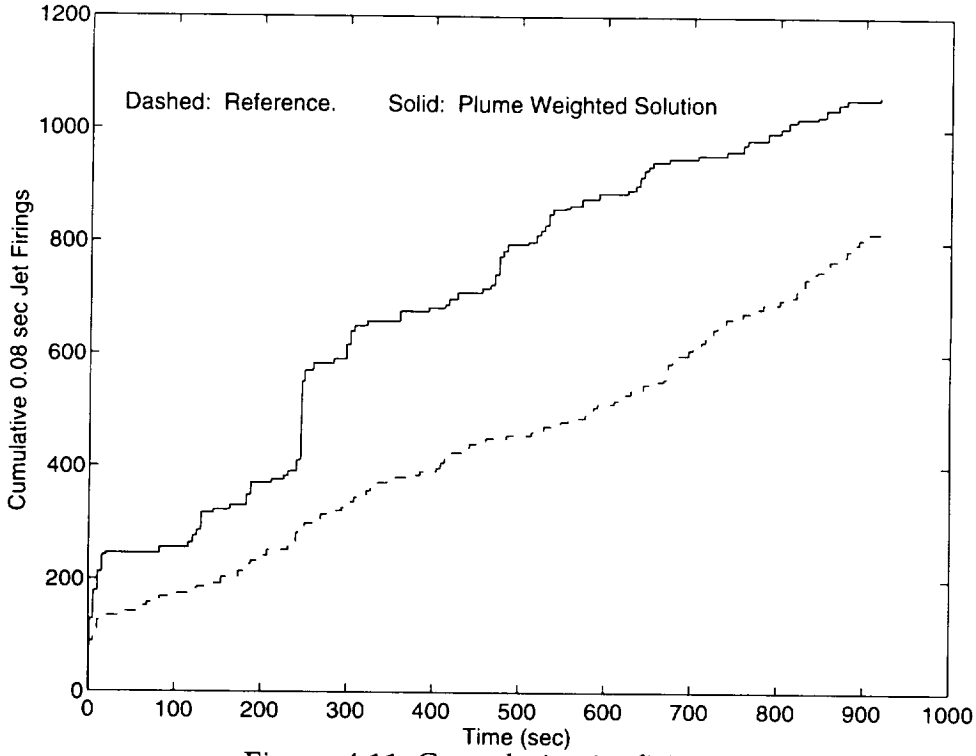


Figure 4.11 Cumulative jet firings.

The cumulative applied force is related to the total impingement cost for the run, but for many structures, the maximum force applied at a given firing may be more important. Thus the large spikes of figure 4-9 may be of more concern than the total impingement - depending on the strength of the structural element.

Figure 4-12b shows the 0.1 psf plume bulb contacting the solar panel during the reference run. It is clear from the figure that the panel is in the heart of the flowfield, where particle density and heat flux are generally highest (see chapter two). Table 4-1 shows the number of times that the 0.1 psf plume bulb contacted the solar panel or the radiator for each run.

	Reference Trajectory	Plume Weighted Traj.
Radiator	3	1
Solar Panel	9	0

Table 4-1. Number of intersections of the 0.1 psf iso-pressure region with the solar panel and radiator.

These tables are presented in the first two runs only since the last two runs are further away from the structure and no contact with the 0.1 psf bulb occurs.

For this close-in validation run, we concentrated on two particular pieces of structure. For all further runs, the force applied to the entire station (SC-05 or SC-24) was analyzed and the weightings of figure 4-4 used in planning.

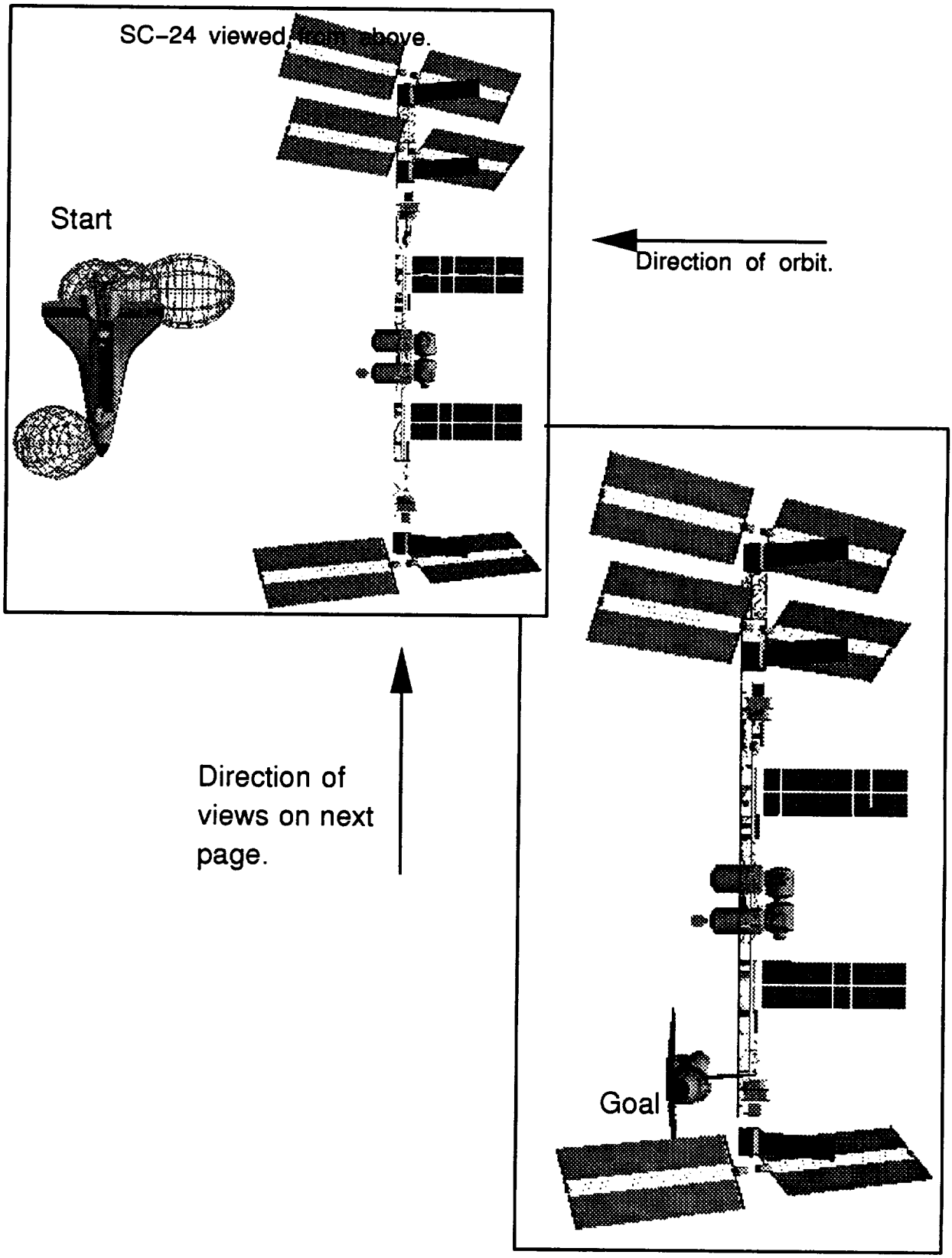


Figure 4-12a. Start and goal positions for case 1.

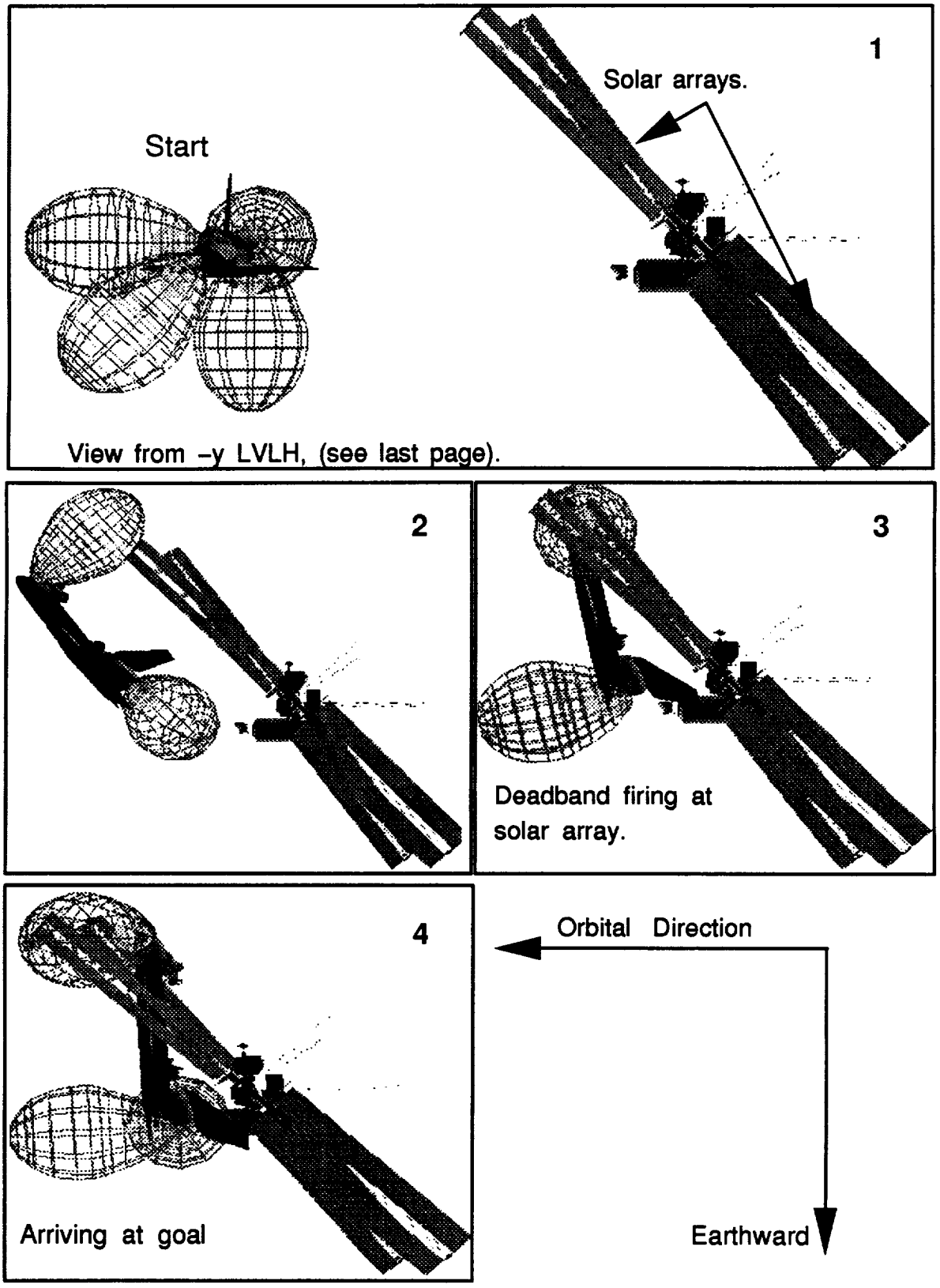


Figure 4-12b. Reference trajectory for case 1 showing solar panel impingement.

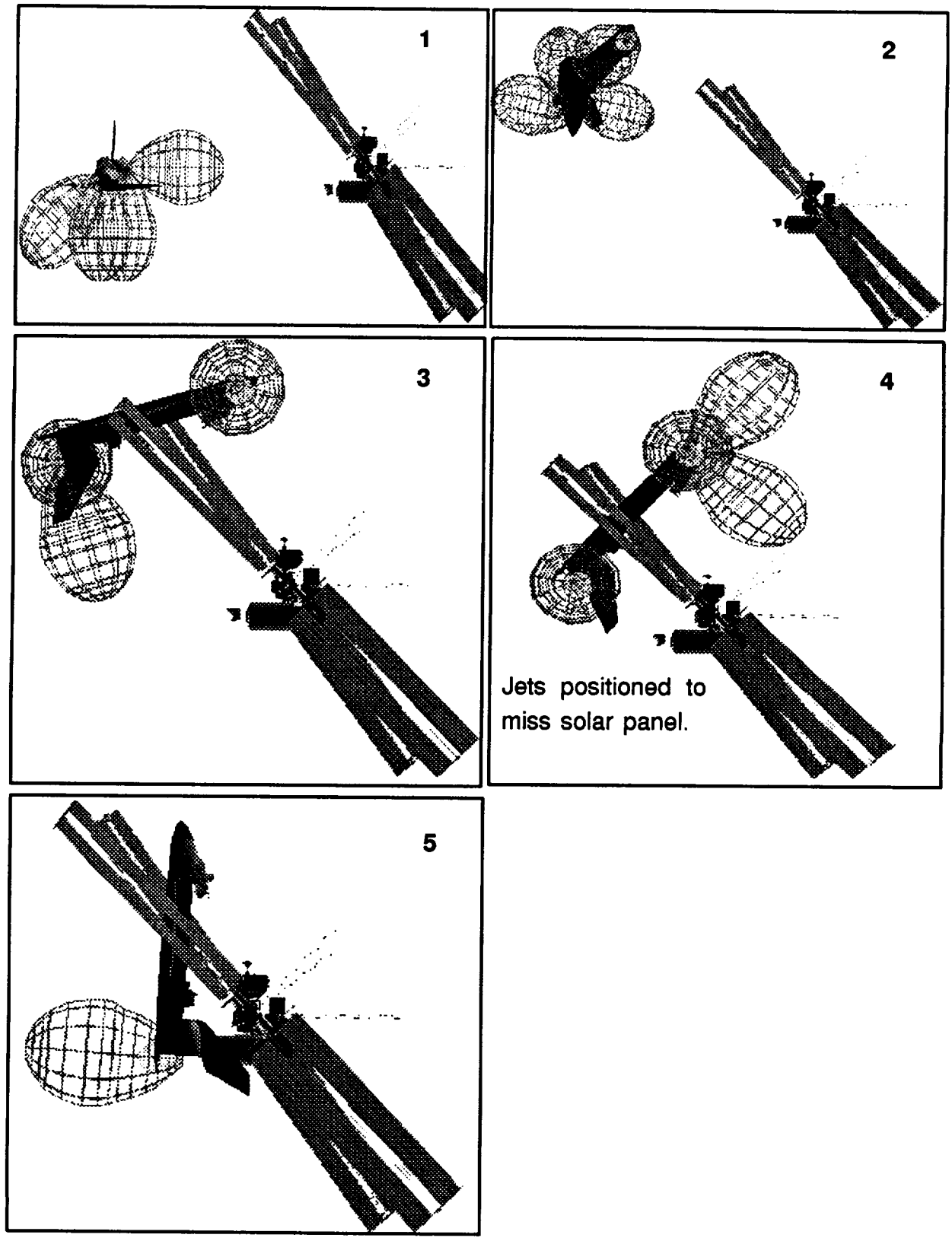


Figure 4-12c. Plume weighted trajectory for case 1 showing positioning of jets.

CASE 2: DOCKING RUN TO SC-05

This run was conducted to examine the handling of constraints and their impact on costs. The start position is in the orbital plane, 200 feet in front of the station (refer to the figures at the end of the section). The objective is to arrive at the docking port while minimizing impingement, with the following constraints (these were fabricated by the author and do not necessarily meet any real world standard):

1) Translation: Inside 10 feet the Shuttle docking port must be within a cone, centered about the space station docking axis. At the docking port, the cone is 6 inches in diameter, expanding to 3 feet at 10 feet out. The closing velocity must be between 0.1 and 0.2 feet per second and the velocity in the y-z plane (shear velocity at the port) must be less than 0.01 feet per second.

2) Attitude: Inside 5 feet, the Shuttle must be at the goal attitude (within deadbands).

The velocity constraints were handled by simply assigning a velocity at the goal. A* then worked backward from the goal, adding perturbed delta v's to this final state to get the previous states. The attitude and translation constraints were handled by logic which simply eliminated nodes that did not meet the constraints by not putting them on the unexpanded heap.

Figure 4-13 shows the translational trajectories planned with and without plume weighting. The docking constraint is clearly visible at the end of each trajectory. Note that the plume weighted solution stays lower and bends out of the orbital plane away from the structure.

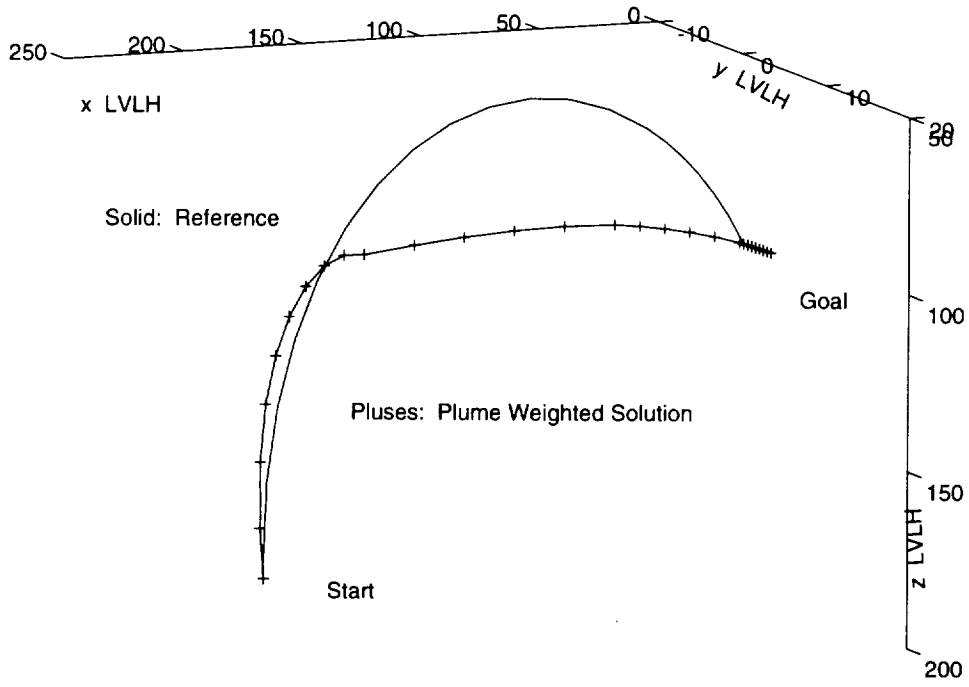


Figure 4-13. Reference solution and plume weighted solution.

Figure 4-14 is the attitude comparison. The Euler angle plots appear complex, but they describe a simple motion. The sum of the plotted angles represents a maneuver in yaw, so vehicle simply "leans" away from the heavily weighted radiator, so that the jets in group 1 (nose jets) and those in group 3 (left side jets) straddled the radiator. This maneuver also reduces docking port and solar panel impingement through most of the trajectory. (See figure 4-19b, frames 3a and 3b.)

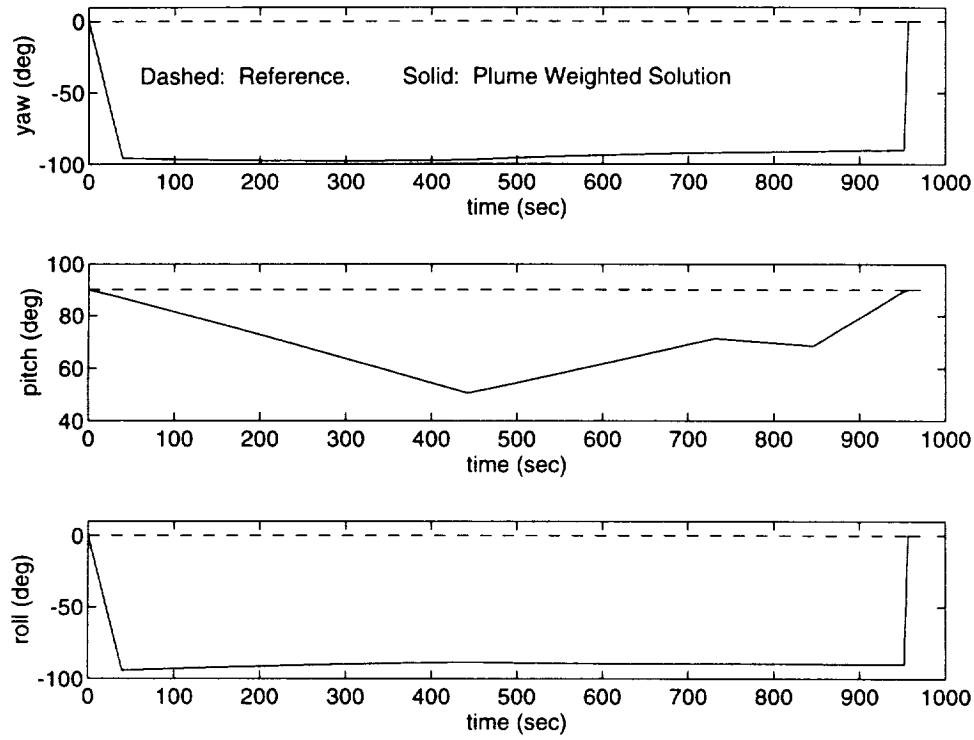


Figure 4-14. Attitude for reference and plume weighted solutions.

The costs predicted for each trajectory are shown in figure 4-15. Note the high peaks in the deadbanding effects function that occur as the reference trajectory passes the radiator. From the solid deadbanding effects curve, we should expect virtually no deadbanding costs for the weighted trajectory until very close to docking.

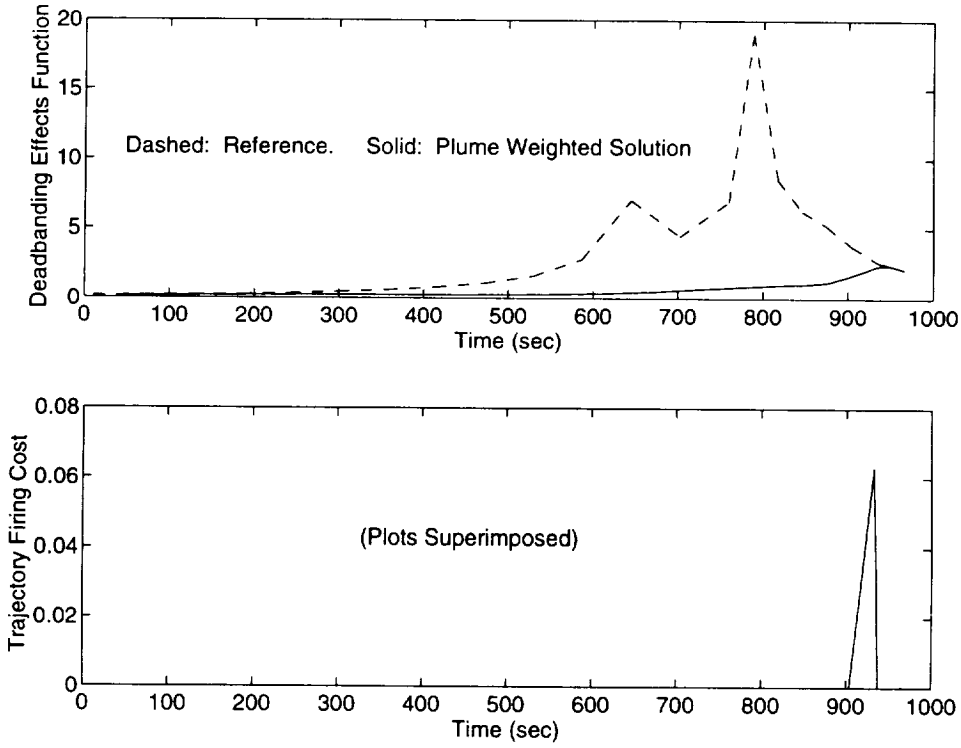


Figure 4-15. Predicted costs for SC-05 docking run.

From the bottom plot in figure 4-15, we can see the effect of the docking constraint on costs. The trajectory firing cost at the goal for both trajectories is virtually the same. This should be expected since the number of optional directions working backward from the goal, is limited by the constraint.

Figure 4-16 shows the applied forces on the entire structure. The spike at the end of the trajectory occurs in both trajectories (the plots are superimposed at

the goal). This corresponds to the predicted trajectory firing cost from figure 4-15. Note that the peaks in the reference trajectory correspond closely to the predicted deadbanding costs of figure 4-15.

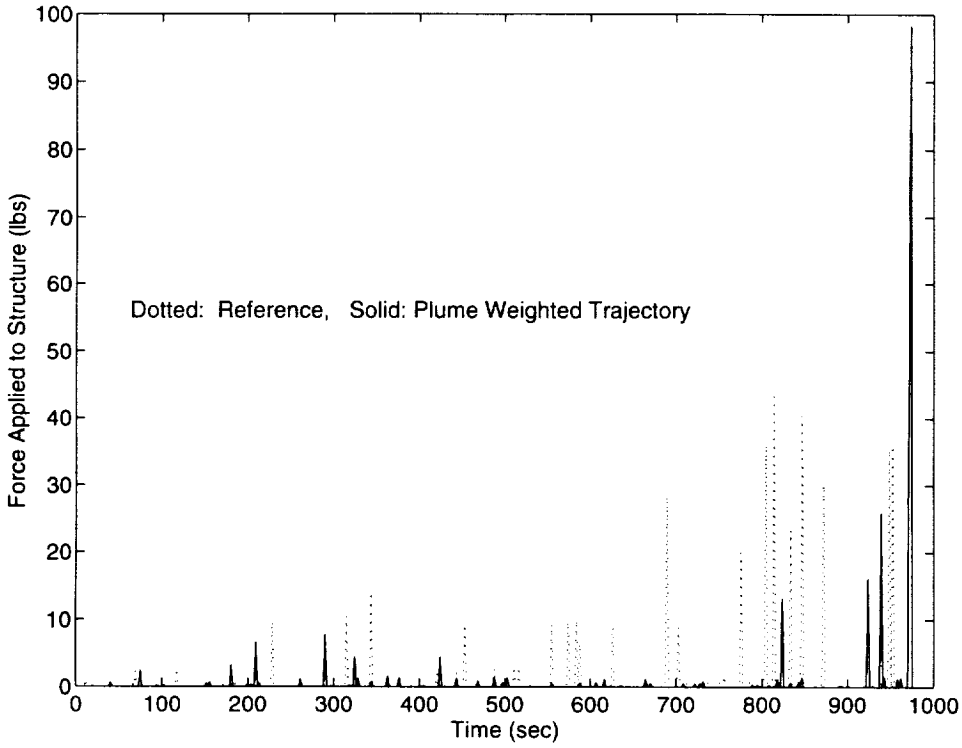


Figure 4-16. Applied force on SC-05 for docking runs.

Figures 4-17 and 4-18 show the cumulative applied force and cumulative jet firings. Here, plume costs are drastically reduced for a fairly small fuel price (with the same tight deadbands mentioned in the last section).

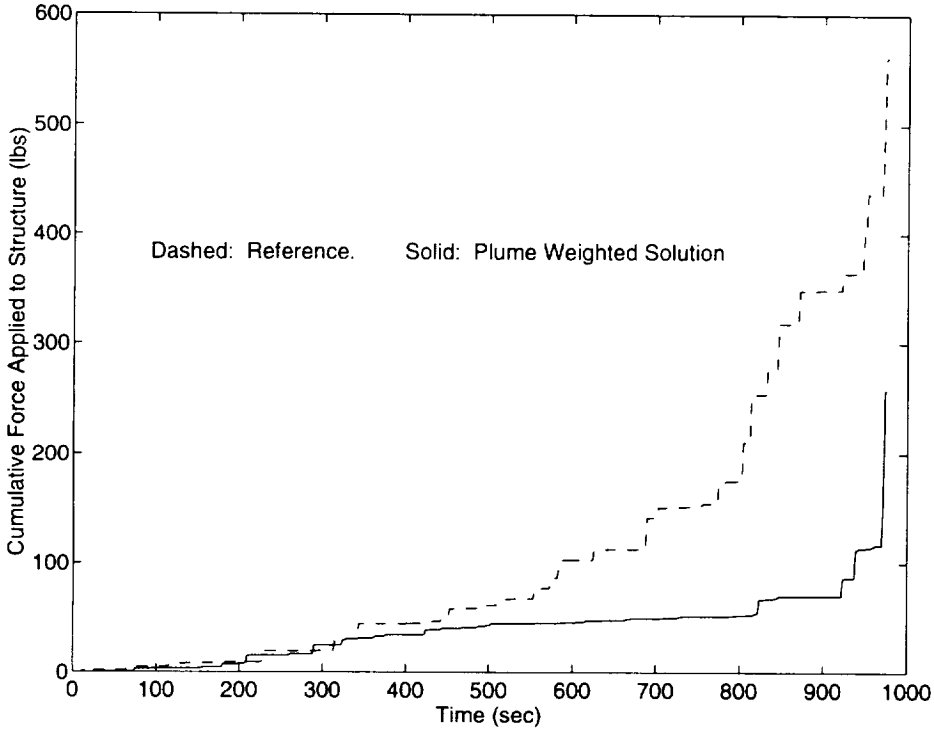


Figure 4-17. Cumulative applied force.

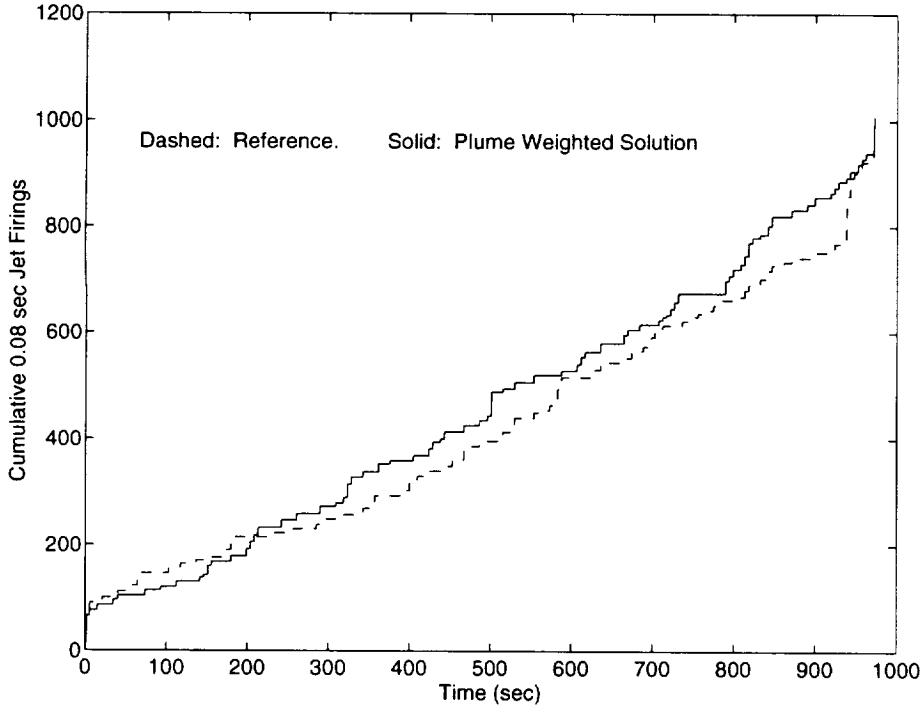


Figure 4-18. Cumulative jet firings.

Cumulative applied force was reduced by 54% at the cost of an 8% increase in total jet firings.

Table 4-2 shows the number of intersections of the 0.1 psf and the 1.0 psf iso-pressure regions with the structure.

	Reference Trajectory	Plume Weighted Traj
0.1 psf envelope	19	5
1.0 psf envelope	3	0

Table 4-2. Intersections of the 0.1 psf and 1.0 psf iso-pressure envelopes.

In cases where the structure is fairly simple, the A* search can often reduce plume impingement dramatically.

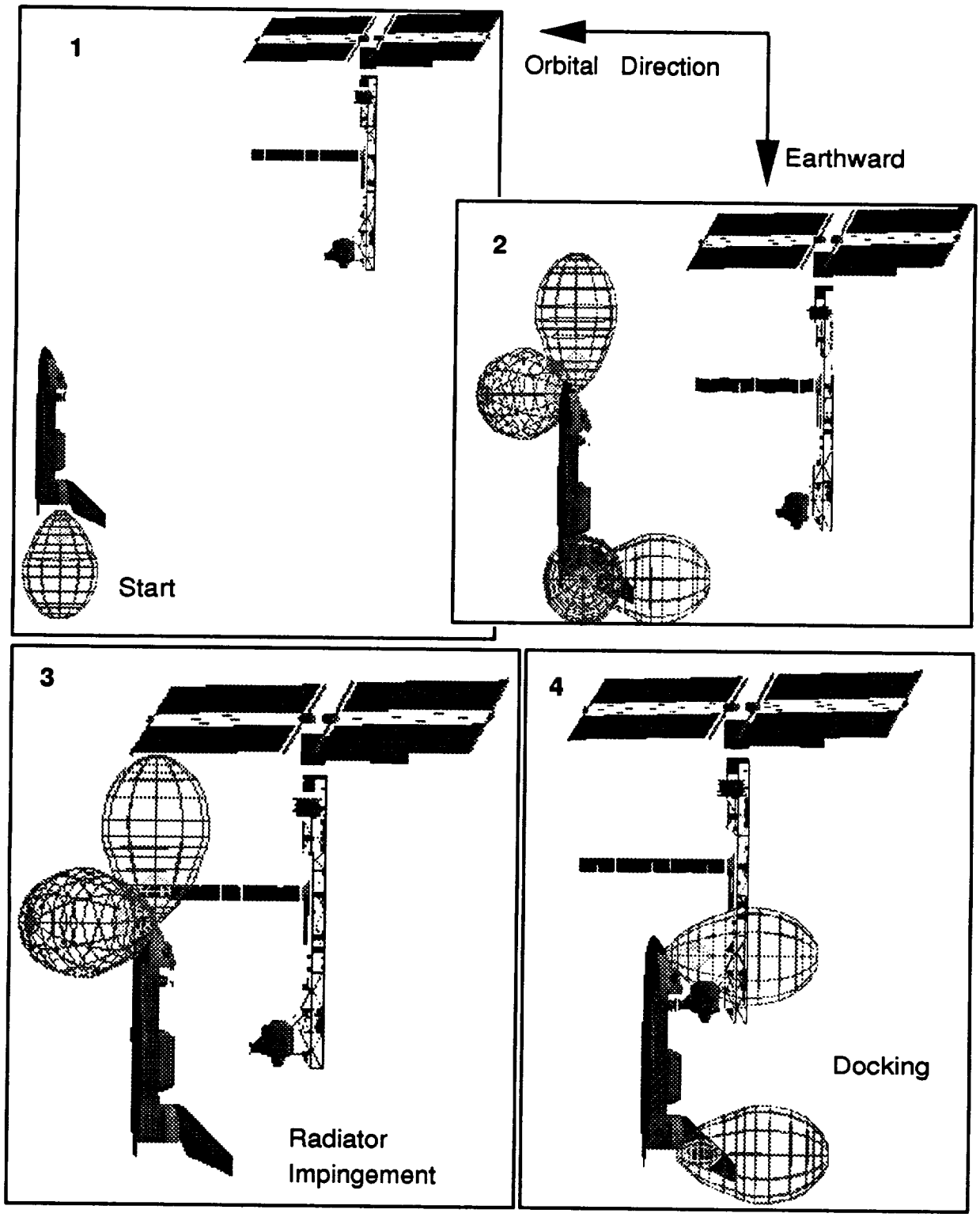


Figure 4-19a. Reference docking trajectory.

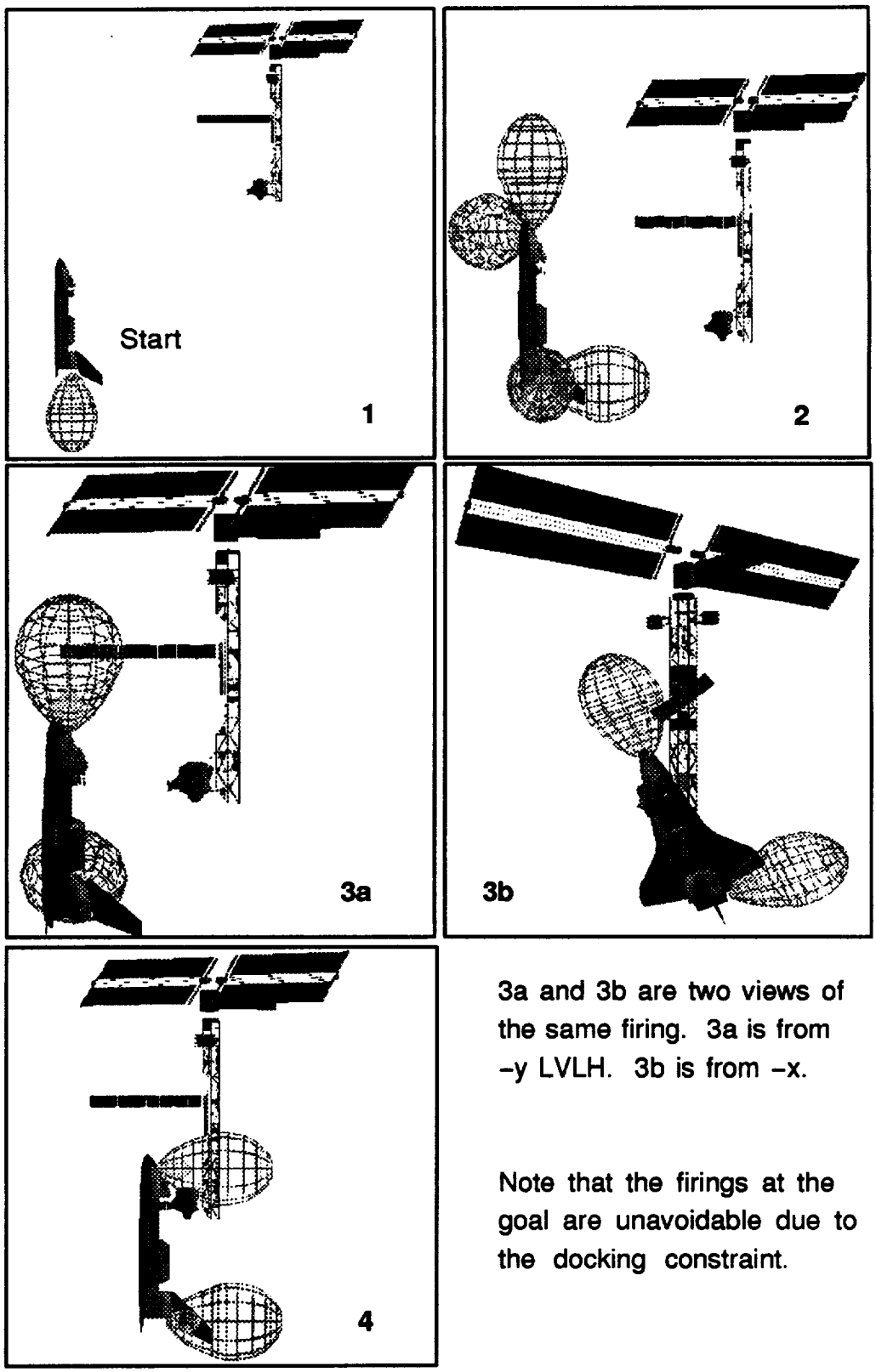


Figure 4-19b. Plume weighted docking trajectory.

CASE 3: R-BAR APPROACH TO INITIAL DOCKING POINT

One possible use for an automated proximity operations controller is to fly an automatic approach to some initial docking fix, and then have the docking performed manually. This case simulates such an approach, starting 400 feet below the station on R-BAR, going to a position 100 feet ahead of the station on V-VAR. The goal position is between the solar panels on the positive x_{LVLH} side of the station (see figure 4-4). Figure 4-20 shows the reference and plume weighted solutions. In translation, it is easy to see how plume costs are reduced. The larger loop increases the average distance from jets to structure through most of the trajectory thereby reducing deadbanding costs, and the arrival velocity at the goal is earthward, so that firings which stop the vehicle will also likely be earthward – away from the target.

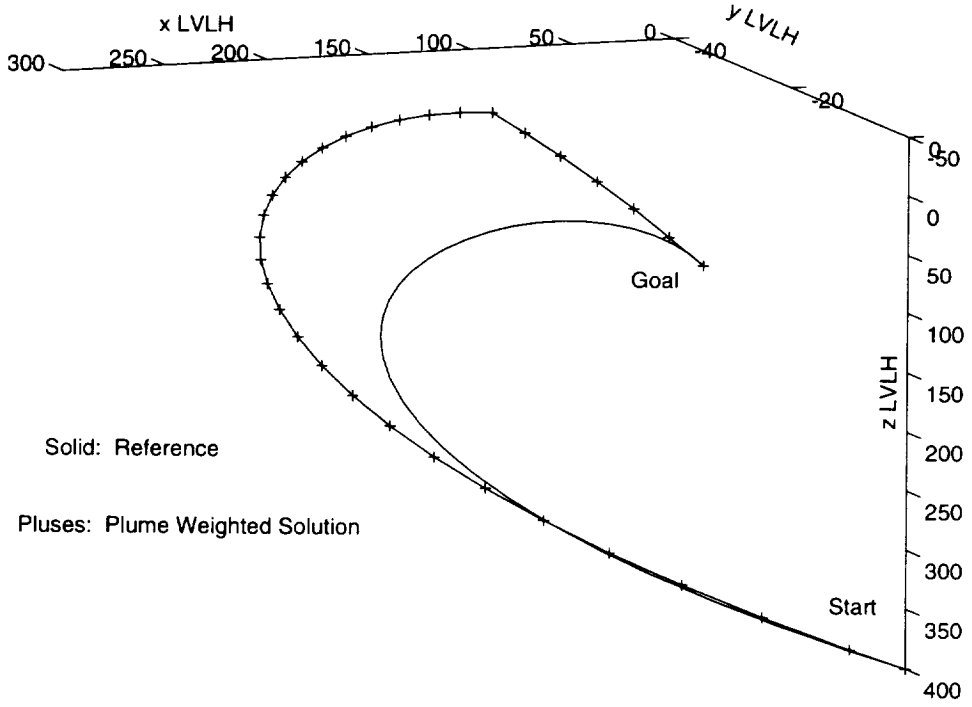


Figure 4-20. Reference solution and plume weighted solution.

Figure 4-21 shows the attitude profiles for both trajectories. These variations in attitude can best be visualized by referring to the simulation pictures (figures 4-26a and 4-26b) at the end of this section. Note that the plume weighted solution approaches the station at a drastically different attitude than the reference. An extra iso-pressure plume bulb (0.01 psf) has been added to the simulation pictures to help compare impingement at the larger average ranges for this trajectory.

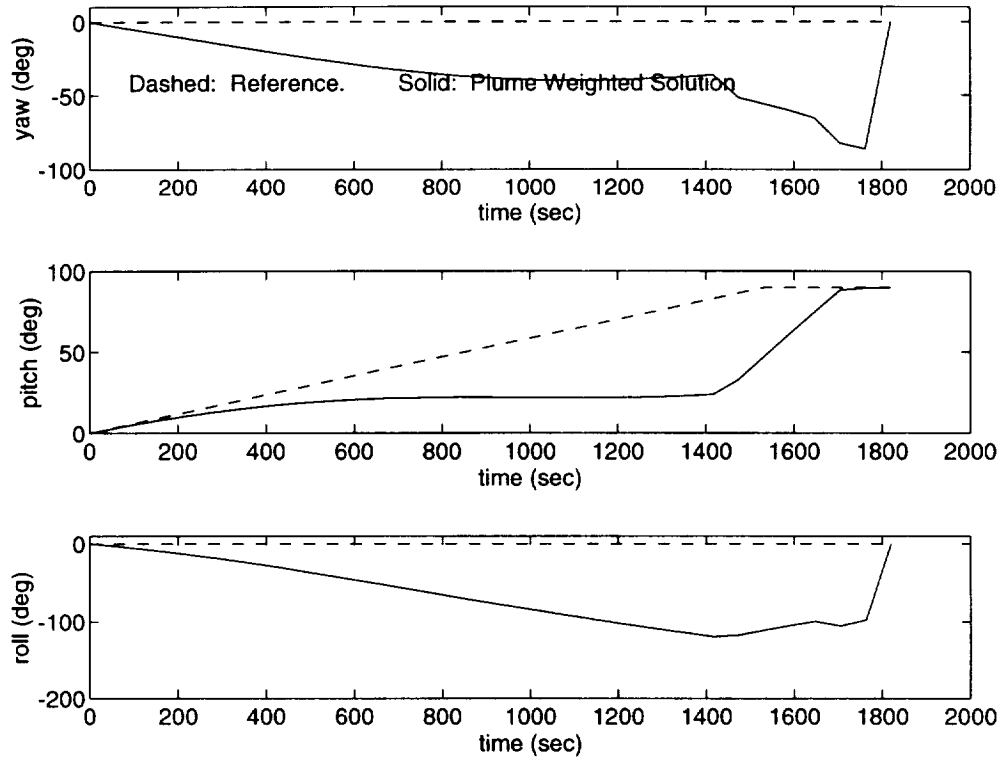


Figure 4-21. Attitude trajectories.

The change in attitude was planned by A* to reduce the deadbanding costs of figure 4-22. Note however that the scale of the deadbanding effects function is lower for both curves than for either of the previous two cases. Thus deadbanding costs should play a smaller role than trajectory firing costs on execution.

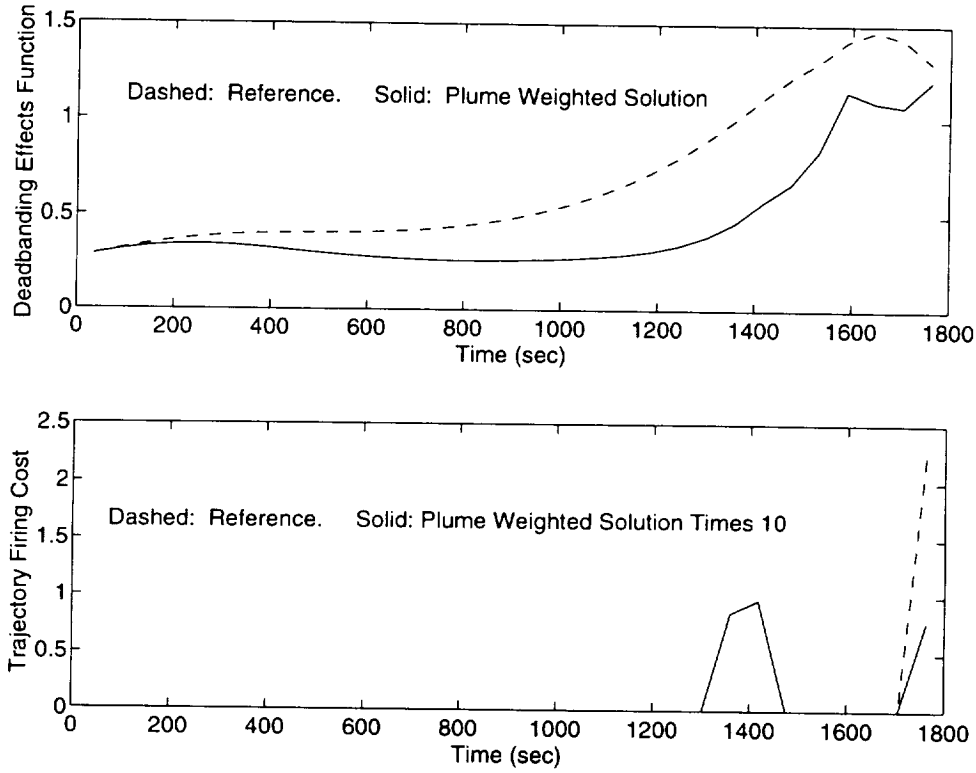


Figure 4-22. Predicted plume costs.

The deadbanding effects function was designed to measure the likelihood of expensive deadbanding firings. However, "unlucky" firings can still occur as evidenced by the spike at about 1550 seconds in the solid curve of figure 4-23. Note that the time of occurrence corresponds roughly to the peak at about 1590 seconds of the deadbanding effects curve (Fig 4-22). Varying initial conditions caused different firing patterns which did not produce this spike, but it is presented in the results as representative. Similar spikes often occurred in the reference trajectory as well. Even though the structure was coarsely modeled as a collection of nodes, the overall cost of deadbanding seems to correspond fairly well with the predicted effects, and the overall applied force was reduced in weighted trajectories in all cases.

A major difference in the cost for stopping at the goal was achieved by changing the direction of arrival. This lower cost is predicted in the firing cost function of figure 4-22 (note that the weighted trajectory solution is multiplied by ten to make it visible). The small spikes at the goal time in figure 4-23 confirm the prediction. Figures 4-26a and b help to visualize this by showing the jet firings that occurred upon stopping at the goal for each trajectory.

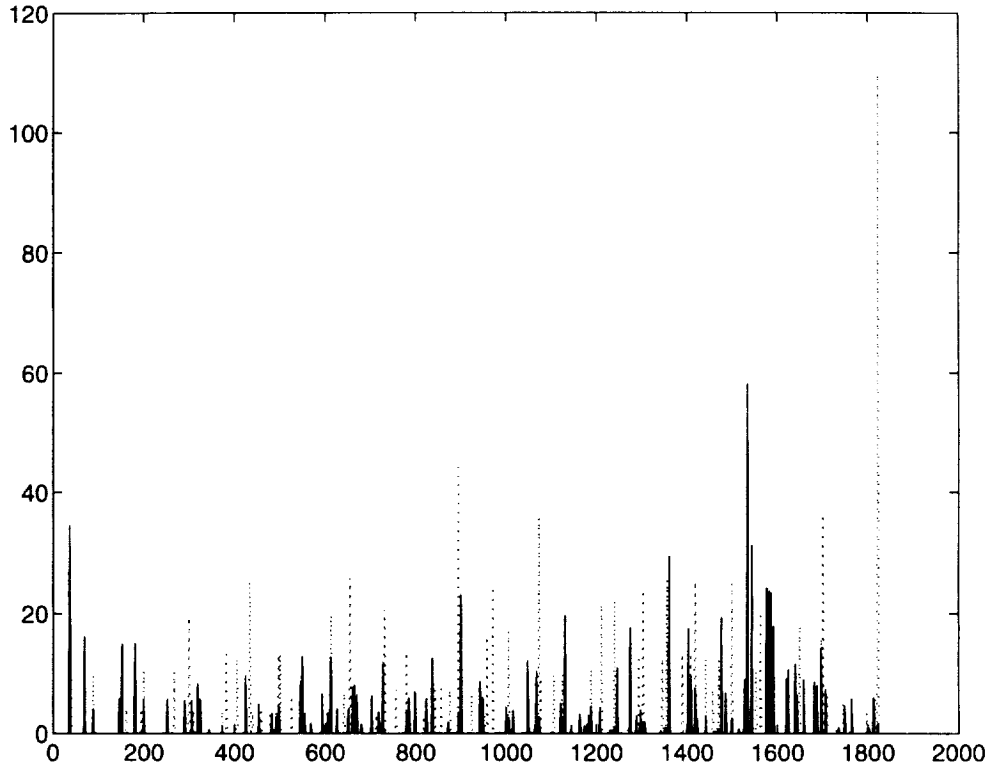


Figure 4-23. Forces applied to the station due to jet firings.

Figures 4-24 and 4-25 show cumulative forces and jet firings. The steep rise in cumulative force in the weighted trajectory at about 1600 seconds corresponds to the steep rise in the deadbanding effects function at the same time.

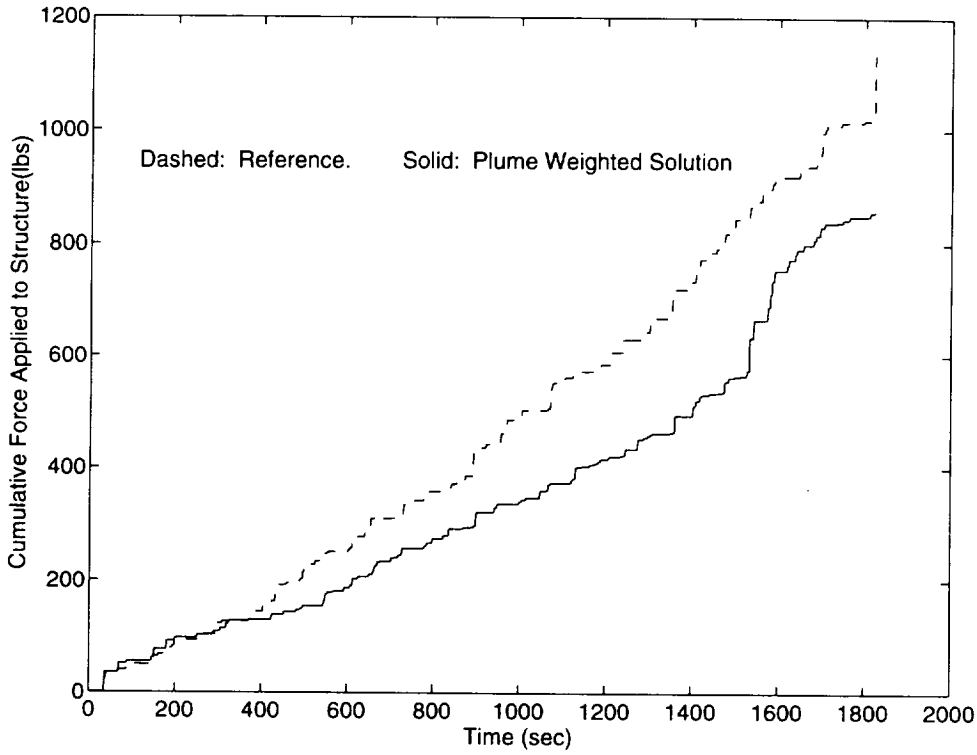


Figure 4-24. Cumulative force applied to structure.

Once again, there is only a small increase in total jet firings (about 5%) with tight controller deadbands throughout.

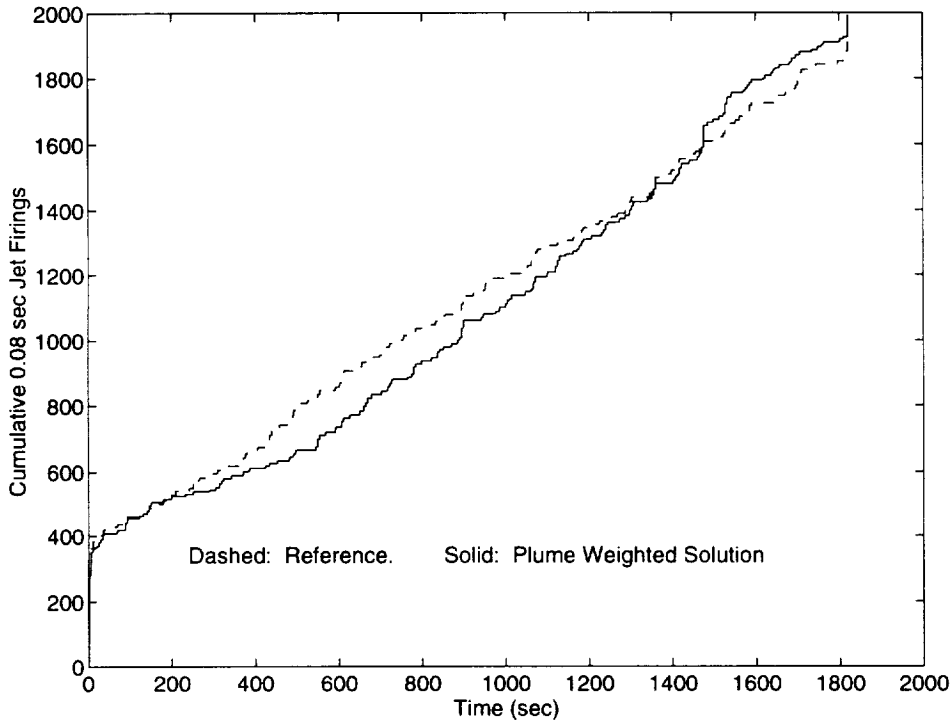


Figure 4-25. Cumulative jet firings.

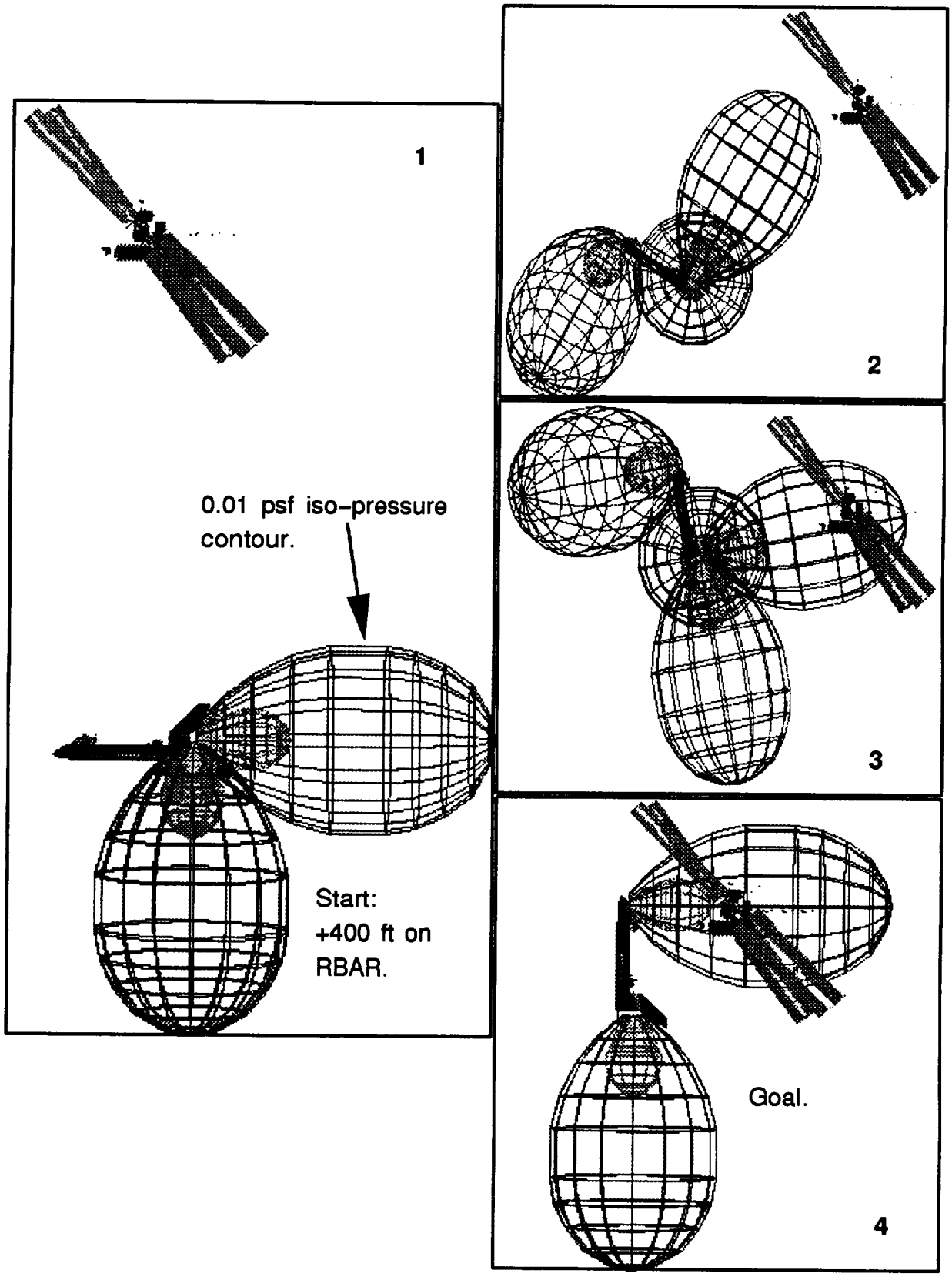


Figure 4-26a. Reference trajectory from 400 ft on R-BAR to initial docking point.

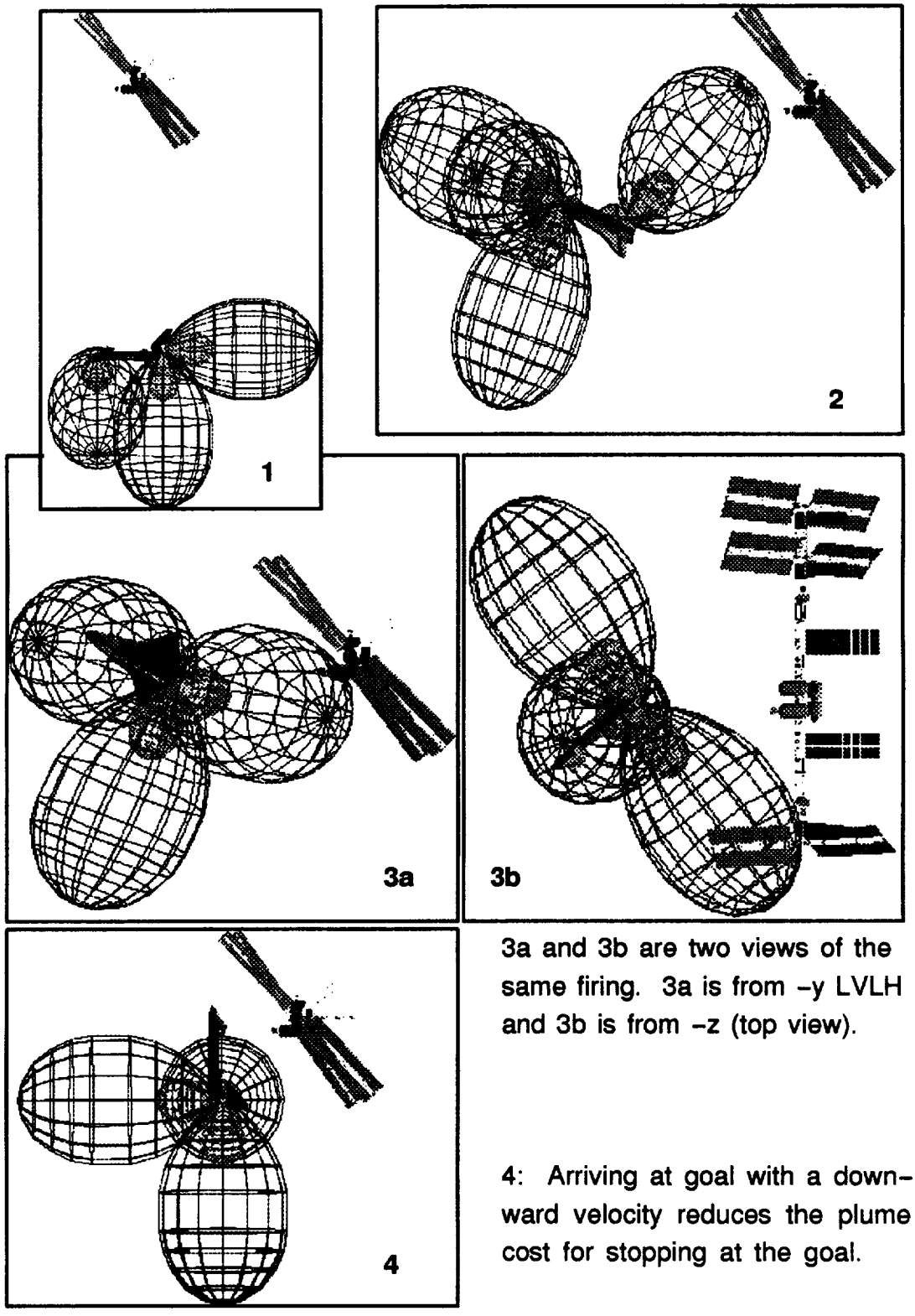


Figure 4-26b. Plume weighted R-BAR approach.

CASE 4: QUARTER FLY-AROUND

A fly-around is a maneuver commonly performed to inspect the target spacecraft prior to docking or grappling. As discussed in chapter three, A* converges best when the fly-around is broken into pieces. A forward or reverse time search – depending on the time-direction of decreasing cost – can then be used to achieve a solution in each piece. This scenario starts with the orbiter at 300 ft below the station on R-BAR. The overall objective is to fly around the station from end to end. This trajectory is one quarter of that circumnavigation. The goal for this piece of the fly-around is 100 feet from the $-y_{LVLH}$ end of the truss, in the tangent plane. Figure 4-33a and b can be used to help visualize the start and goal states. The views in these figures are from the $-x_{LVLH}$ axis (behind the station in the orbital plane).

Figure 4-27 shows the weighted solution which is superimposed on the reference solution in translation. The figure underscores the non-intuitive nature of space trajectories. Even though the start and goal states are both in the plane defined by x = 0, the fuel optimal trajectory flies well out of this plane (to x ≈ 150 ft) in order to account for the x-z coupling discussed in chapter two. Thus, arrival velocities at the goal are not what one might first expect. Instead of arriving with an upward (-z) velocity component, the vehicle arrives with a sizable drift in the $-x$ direction. The size of this "loop" into the positive x direction is a function of the desired time of flight between the two states.

The fuel optimal solution already has many characteristics which are desirable for impingement reduction. The trajectory arcs far away from the structure which reduces deadbanding costs. It also arrives at the goal with a

"plume-cheap" velocity, meaning that the jets required to stop at the goal do not impinge the station. For these reasons, the A* solution does not change the translational trajectory at all and the weighted solution differs from the reference only in rotation (Fig. 4-28).

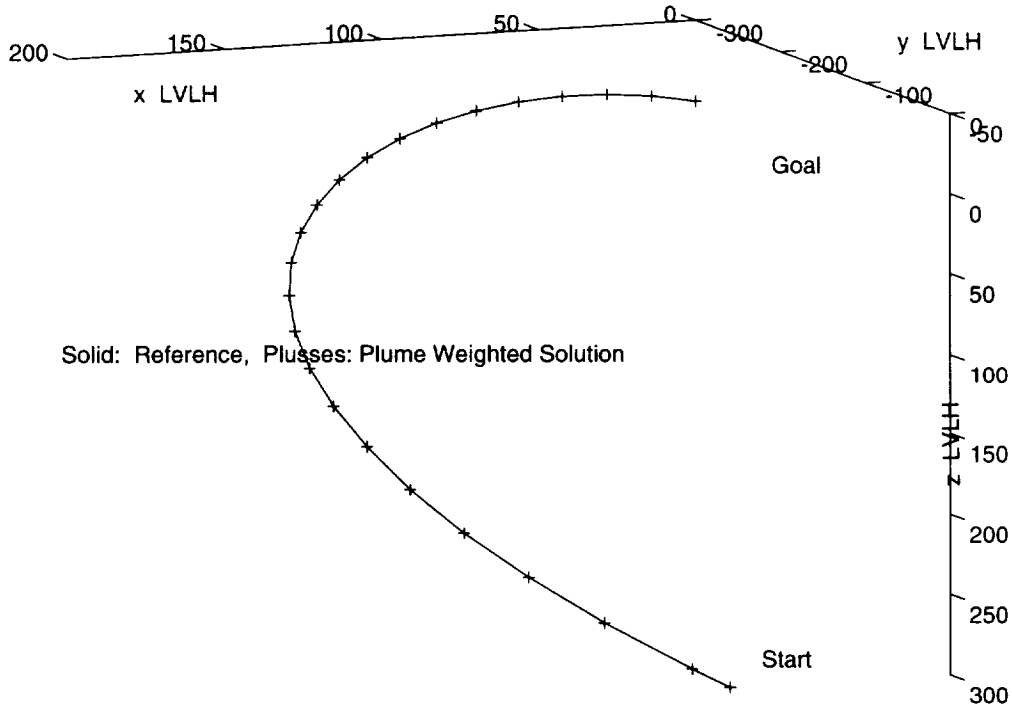


Figure 4-27. Reference and plume weighted trajectories.

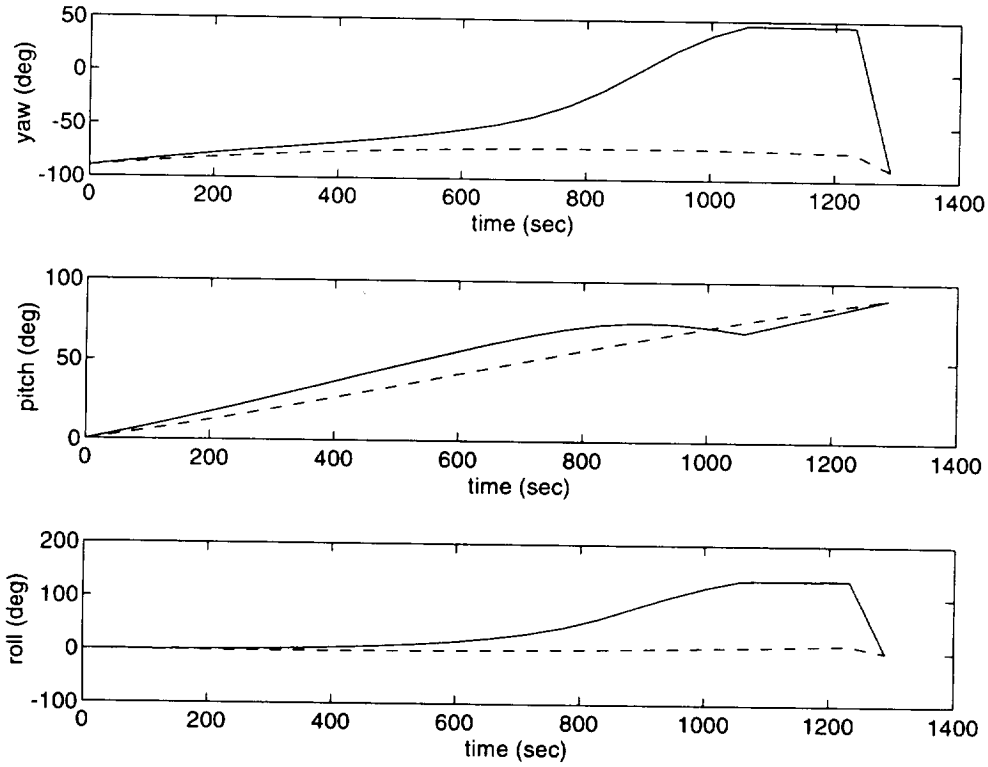


Figure 4-28. Attitude trajectories.

Figures 4-33a and b help to visualize the attitude differences between the reference and weighted solutions. From the $-y_{LVLH}$ direction (Fig. 4-33b, frame 3b), we can see that the weighted solution maintains a significantly different attitude when in the vicinity of the solar panels at the end of the structure. This reduces the impact of deadband firings. Note that this reduction is predicted by the top plot of figure 4-29.

The plume costs for stopping at the goal (bottom plot of figure 4-29) are virtually the same for both trajectories. This is because the arrival velocities are the same in translation and the differences in rotational arrival velocities do not affect the stopping cost much (in this case).

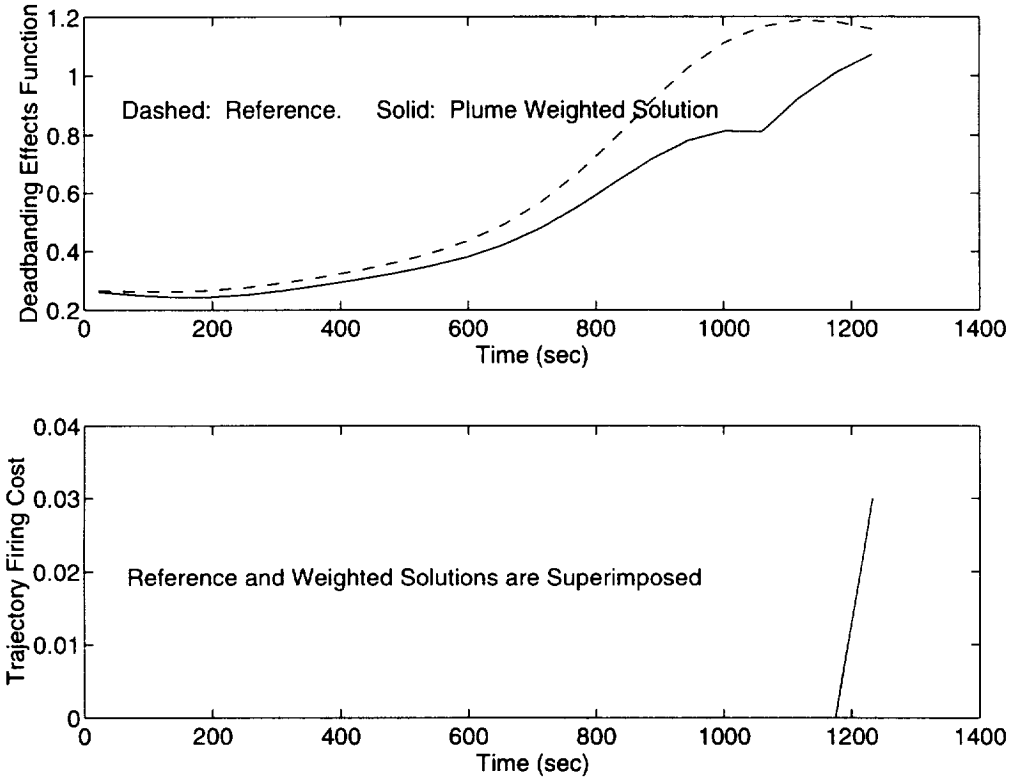


Figure 4-29. Predicted plume costs.

Figure 4-30 shows the forces applied to the structure at each jet firing. Notice that both trajectories produce a spike of force during the start firing. These spikes occur from the large number of firings required to start the vehicle on its path, even though most of the jets fired at the start were directed away from the structure. As mentioned above, the force spikes are the total force applied while actuating a velocity change, so they are not as impulsive as they

appear. The rest of the applied forces generally correspond to the predicted deadbanding effects of figure 4-29.

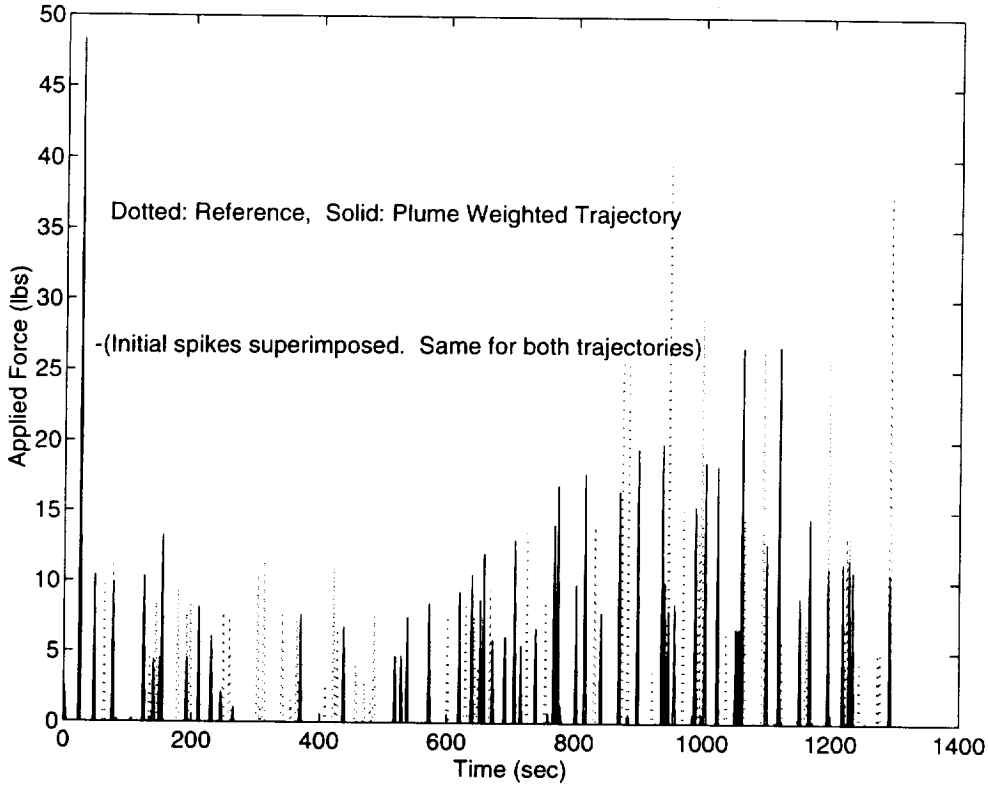


Figure 4-30. Forces applied to the structure due to jet firings.

Figure 4-31 shows the cumulative force results. The improvement here is on the order of 10% because the reference trajectory has many favorable characteristics for plume impingement reduction.

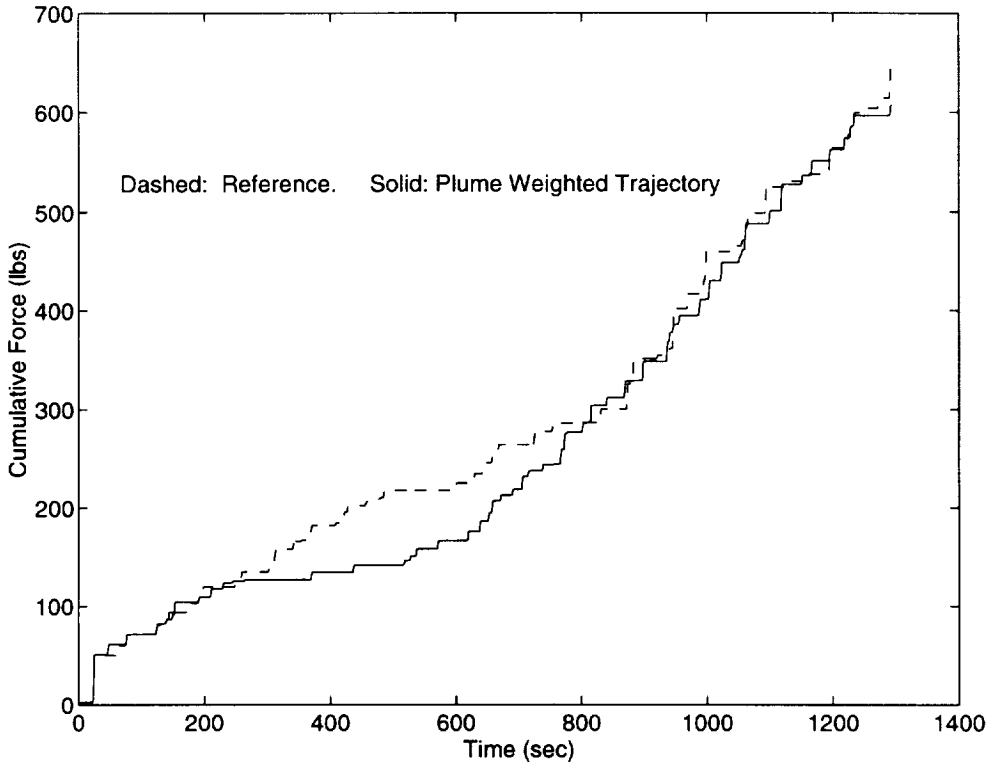


Figure 4-31. Cumulative forces applied to structure.

Since the two trajectories are the same in translation, the fuel costs can be expected to be similar. Figure 4-32 confirms this, showing about a 10% increase in jet firings to accomplish the planned rotational maneuvers.

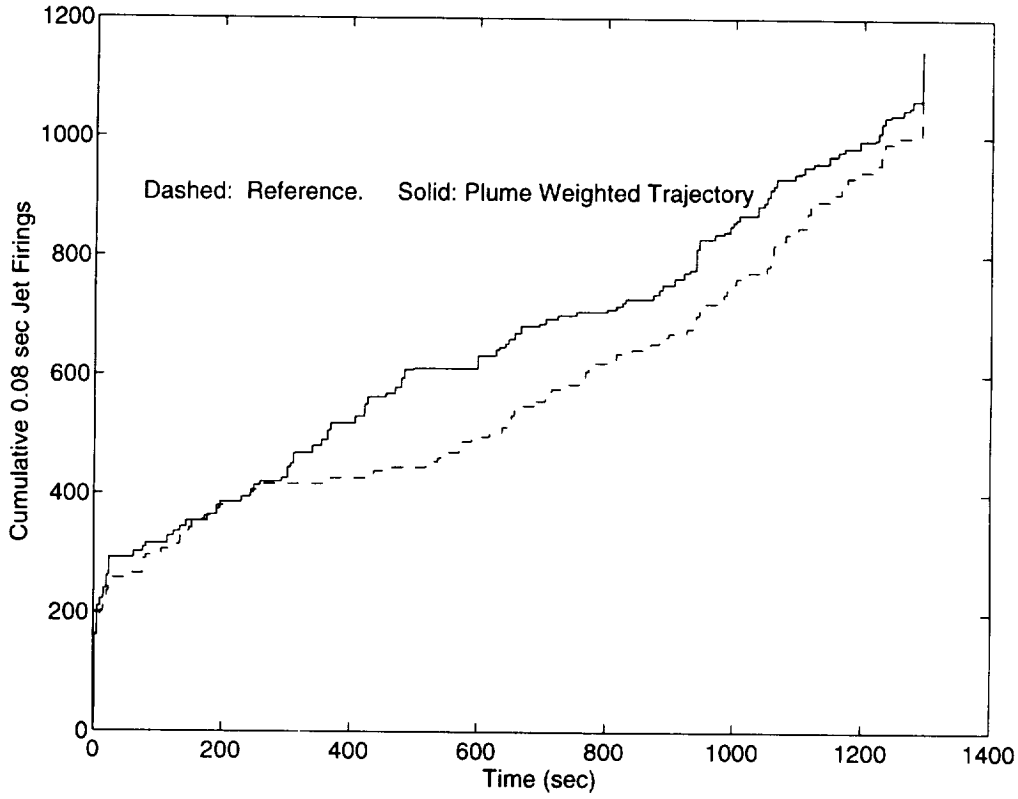


Figure 4-32. Cumulative jet firings.

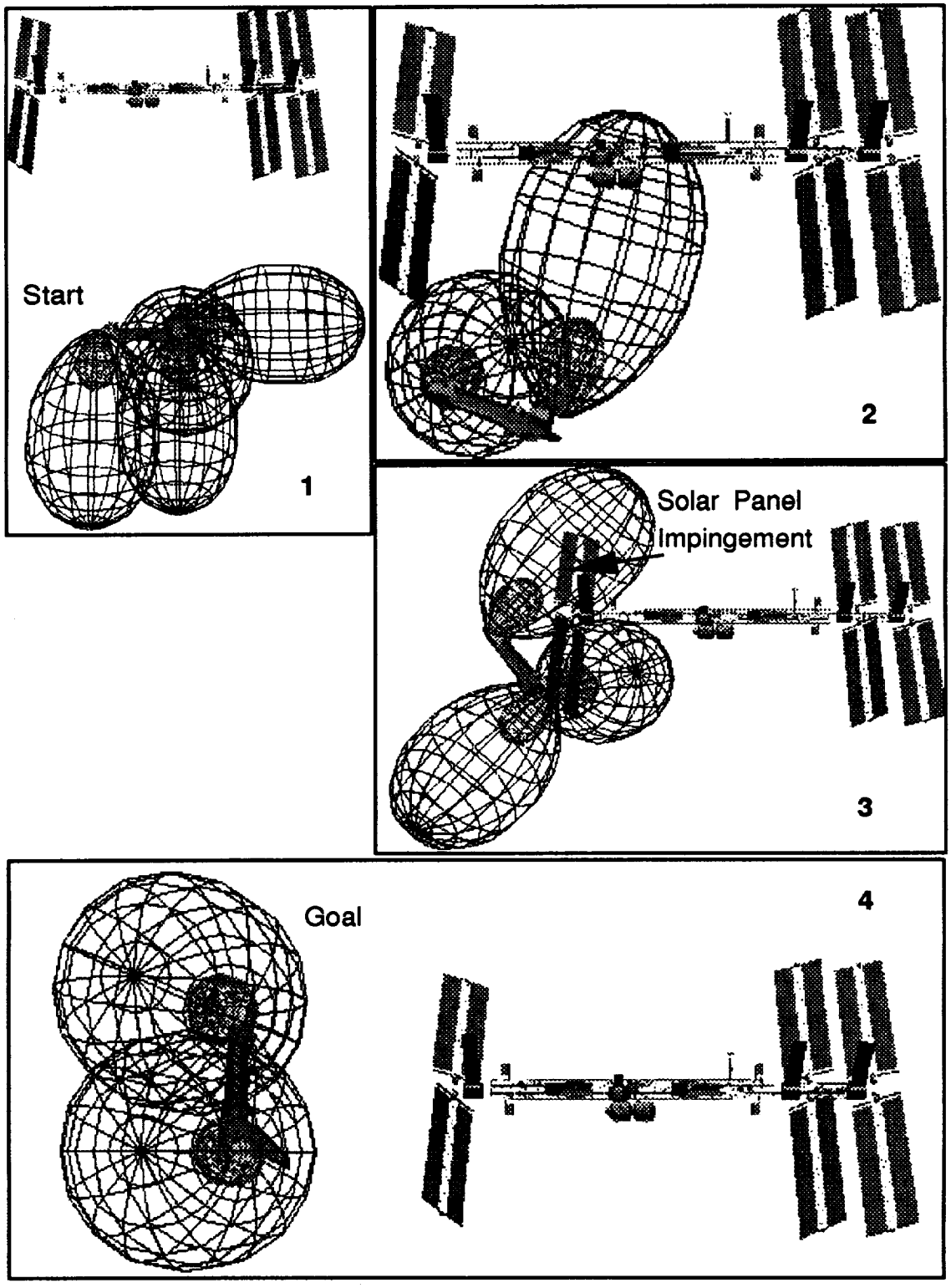


Figure 4-33a. Fly-around: reference trajectory.

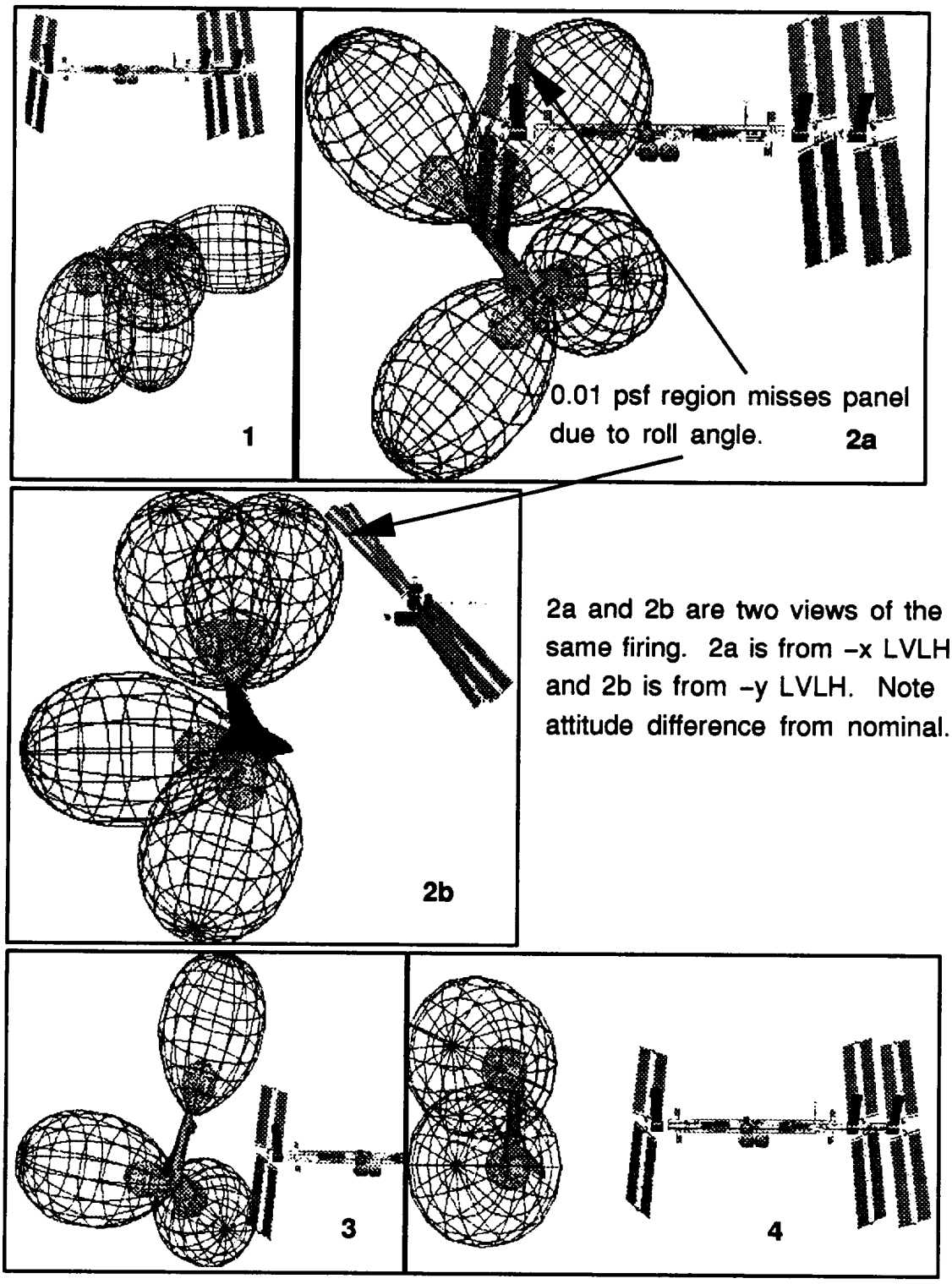


Figure 4-33b. Fly around: plume weighted trajectory.

COMMENTS ON A* CONVERGENCE

Two techniques were used to improve the convergence of the algorithm. A dynamic weighting factor (see chapter three) of 0.2 was used for all runs, and the time between waypoints was varied as a function of distance to the structure. The time steps varied from 5 seconds (used on docking runs at ranges less than 20 feet) to sixty seconds far from the target. The run time of the algorithm on a Sun Workstation varied between about 45 seconds to ten minutes depending on the length and complexity of the trajectory, the size of the target structure and the distance of the goal from the structure.

CONCLUSIONS

CHAPTER FIVE

The properties of A* guarantee that the trajectories presented are ε-optimal (see chapter three) for the given cost function over the discrete space searched. The next two sections examine the cost function and discretization schemes used here, in light of the results of chapter four. Next, some suggestions for improving the algorithm and some recommendations for future research are presented, and lastly, the main conclusions of this thesis are summarized

COST FUNCTION EVALUATION

The cost functions developed in chapter three differ from the applied forces generated by ICDS in four ways. First, the nodal approximation appears more granular at shorter ranges, which means that the deadbanding cost function will be have "bumps" created when flowfields pass the nodes which approximate continuous structural elements. Second, the nodal approximation allows subjective weighting of structural members. This is clearly important since dynamic pressure on rigid truss members will have an entirely different effect than on sensitive solar arrays or radiators. Third, the determination of the plume cost vector **p**, does not account for the orientation the structural elements. Firing at structures which were broadside to the blast was no more expensive in planning, than firing at feathered structures. Finally, the cost model for individual jet firings was simpler than the flowfield model used by ICDS. For these reasons, we cannot expect perfect correspondence between the shape of the deadbanding curves and the height of the force spikes produced during

execution. However, the results did show similarities between the predicted deadbanding and trajectory firing curves, and the applied force plots produced by ICDS. When predicted deadbanding costs were high, the magnitude and/or density of the applied force spikes increased and vice versa. Also, the number of intersections of the 0.1 psf and 1.0 psf iso-pressure curves with the structure was reduced significantly in cases 1 and 2. By virtually any measure, plume impingement was reduced in all four cases. We may conclude then, that our definitions of plume cost directed A* in the right direction, and that the above approximations are reasonable.

The large force spikes that occurred at the start of some trajectories were unexpected. These spikes occurred mostly on R-BAR trajectories where initial firings tended to be horizontal rather than directly away from the station (see chapter four). The ICDS plume model produced residual dynamic pressure at large angles from the thrust axis after long firings, especially when multiple jets from a single group were fired. This is not modeled well by the plume cost function, F, of chapter three. Preliminary testing seems to indicate that most reasonable trajectories between the same start and goal states have about the same firing costs at the start. Figures 4-23 and 4-30 both show these starting spikes and in both cases the reference and weighted trajectories produced nearly equal applied forces at the start. More study is needed to determine whether the plume cost for leaving the start node varies significantly with moderate variations in the direction of departure.

The main focus here was on plume costs so fuel conservation took a back seat both in planning and in implementing a trajectory following controller. See

"Future Work," below for some recommendations on improving the quality of the solution for fuel consumption.

In all cases, the deadbanding and trajectory firing cost functions did direct the A* search in a direction which reduced plume impingement in the results. However, performance may be improved with more accurate cost functions perhaps at the expense of convergence speed.

NODE EXPANSION EVALUATION

No discretization process will cover all possible avenues through a state space. It is interesting however, to see if testing results illuminate ways in which the process may be improved. The primary assumption that drove the node expansion strategies of chapter three is the fixed time of flight assumption. Reference 6 allowed variations about a nominal time of flight in order to reduce fuel consumption. Such variations might also have a beneficial effect on plume avoidance. As was alluded to in discussing the fly-around case in chapter four, time of flight affects not only the speed at which the trajectory is flown, but also the overall shape of the trajectory, which in turn affects the degree of plume impingement. An example from the testing is the fly-around trajectory. Longer flight times caused the Shuttle to drift far away from the structure, while shorter times caused it to take a more direct route which passed closer to sensitive solar panels.

The use of dynamic node expansion allowed the search to be centered around a nominal coasting trajectory which has desirable characteristics for both fuel and plume costs. This technique suffers from one drawback however – the

number of possible nodes generated is very large. If C is the number of children generated at each parent, then each generation has C times as many possible nodes as the previous one. Thus, if there are n time steps from the start to the goal, then the total number of possible nodes is C^n. Fortunately, A* uses heuristics to prune some of these nodes from its decision tree, but if the heuristics are conservative and the costs are relatively constant, the search can explode and convergence may not occur. Two things can remedy this: improved heuristics, and planning forward or backward in time so that costs generally decrease as the search progresses. Unfortunately, no good method was found for accurately estimating *minimum* plume costs given a particular current position and velocity. Therefore, we have concentrated on backward time searches for approach trajectories. In "future work" below, some ideas are presented for improving the node expansion process.

RECOMMENDATIONS FOR FUTURE WORK

This section contains recommendations for improving the planning algorithm itself, and some ideas derived from testing which may help reduce plume impingement in general. The code used to plan these trajectories is modular in design so that improvements to a particular piece can be made without redesigning the whole.

Cost determination. It may be beneficial to include a normal vector for each of the structural nodes used to represent the target. Plume costs could be made to depend on the angle of incidence between the exhaust and the structure. Of course this would create an additional computational burden, but the effects on

cost determination and convergence speed should be examined so that the tradeoffs are clear.

The computational speed of a lookup table may make it feasible to use the jet select algorithm for fuel cost determination which would improve the quality of the solution for fuel consumption. This would entail running the jet selection routine three times for each child generated at a node expansion. The first selection would determine the fuel generation cost accrued in passing from parent to child (this firing is already done to find trajectory firing plume costs so no additional computation would be required). At each child the jet selection routine would then be run twice more to find the heuristic cost estimate which would consist of firing jets at the child to embark on a nominal trajectory to the goal, then firing them again to stop at the goal.

Node Expansion. Solution quality could be improved by increasing the density of the state space partitioning. In translation, velocity magnitude perturbations could be added which allow more choices at a given node (Fig. 5-1) and rotational maneuver rates could include more options than the 0.2 degrees per second perturbations used here. Of course, increased options at a given node means increased computational burden. One possible compromise would involve changing the node expansion process at each generation of nodes. For example, the inner cone of figure 5-1 could be explored at one generation, and the outer cone explored at the next. This was already done to some extent with the attitude expansion scheme of chapter three which checks the time of the parent and only expands it in attitude if it falls on a multiple of 60 seconds (see chapter 3).

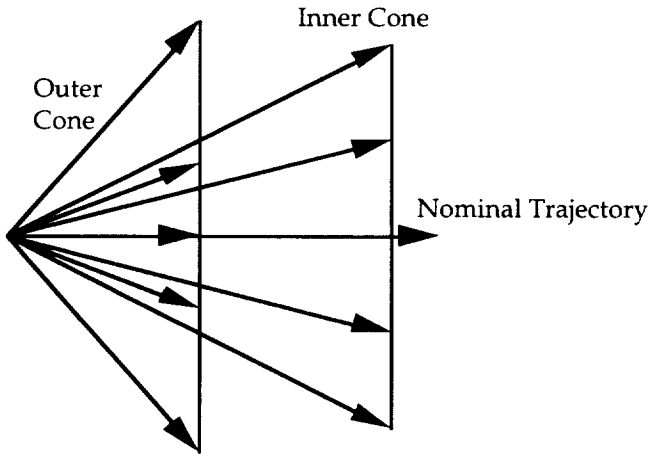


Figure 5-1. Increased options in translational node expansion.

Allowing for flight time variations adds another dimension to the state space partitioning. This would significantly change the node expansion process, but could pay dividends in the quality of the solution for both fuel consumption and plume impingement for the reasons discussed earlier.

Incorporation of collision detection. Collision detection needs to be incorporated to make the planner a useful space-based tool. References 6 and 10 describe proposed collision detection schemes.

Dynamic plume avoidance. Reference 9 details dynamic plume avoidance schemes which use target structural information, together with current state information, to find jet combinations which achieve the commanded rates while reducing plume impingement. A simple example of this idea is the Shuttle's "Low-z" mode which uses the jets in the nose and tail (Groups 1, 7 and 8 in figure 4-1) to produce a positive z rate change. These jets are canted slightly upward so that a $+z$ rate can be produced by firing the nose and tail jets simultaneously. Of course, this is highly inefficient in a fuel sense. Automated

plume avoidance schemes search through possible jet combinations weighing plume costs against fuel costs and command following performance. A combination of plume weighted trajectory planning and dynamic plume avoidance may produce better results than either technique alone.

Controller improvements. The controller that executes trajectory-following commands can play an important role in causing or avoiding plume impingement. The controller used for this project assumed that tight deadbands would be required for close proximity operations. This caused a large number of deadband firings that were not necessary at longer ranges. A controller with dynamic deadbands which change as a function of range, or some other criterion, would improve fuel and plume performance.

Incorporation into an automatic proximity operations package. As manned and unmanned proximity operations become more complex, automated proximity operations will become more commonplace. This A* trajectory planner could fit into the three-tiered system of figure 5-2. At the highest level, a maneuver manager would monitor execution and determine when re-planning is necessary[6]. The A* planner would take desired start and goal states from the maneuver manager and pass waypoints to a feedback controller which would cause the vehicle to move along the trajectory. In manned spacecraft, the maneuver manager may be a human operator.

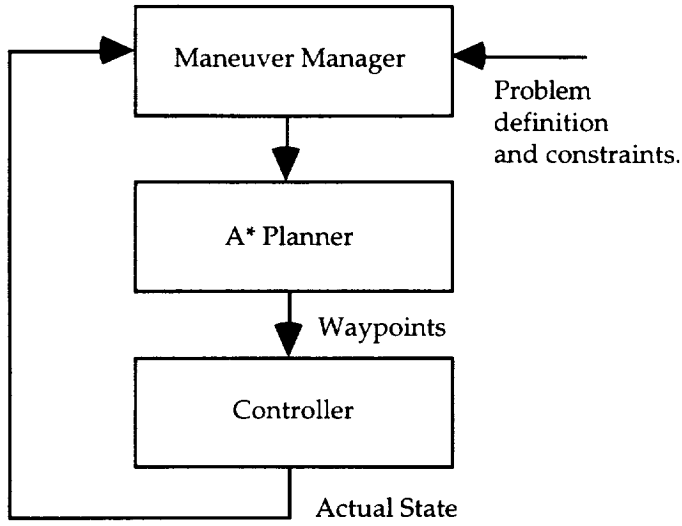


Figure 5-2. Proposed multi-layered proximity operations package.

SUMMARY

This thesis has investigated the use of the A* algorithm to plan plume-fuel optimal trajectories. This section concludes by briefly reiterating the main findings.

A computationally efficient plume cost function was presented which calculates the relative cost of firing a single jet at a target structure represented by a collection of weighted nodes. This function approximates iso-pressure curves within the ranges of interest.

A distinction was made between two types of jet firings that can occur during execution of the planned trajectory. Trajectory altering firings produced fuel and plume costs whenever a nominal coasting trajectory to the goal was embarked upon, stopped or departed. Deadbanding firings are a product of

feedback control schemes and were considered random due to modeling errors, disturbances, and the difficulty of determining precise initial conditions.

The plume cost function was applied to define the relative costs of various candidate trajectories with trajectory altering firings and deadbanding firings considered. The cost for trajectory altering firings was calculated by running the jet select algorithm, and summing the costs for firing each jet. Deadbanding costs were determined by assigning a cost to each jet based on the degree of impingement expected if that jet was fired (see chapter three).

A* convergence is much faster when costs generally decrease with depth in the decision tree. This motivated the use of a reverse time search for approach trajectories. The translational and rotational dynamics of chapter two are easily converted to reverse time and the cost functions of chapter three remain the same.

Finally, the cost functions and node expansion strategies used in this thesis enabled the A* algorithm to significantly reduce plume impingement. The trajectory planner presented here would be useful as a ground-based or space-based tool for manned or unmanned vehicles.

REFERENCES

1. Clohessy, W.H. and R.S. Wiltshire, "Terminal Guidance System for Satellite Rendezvous." *Journal of Aerospace Science,* Sept. 1980: 653-658, 674.

2. Legostayev, V.P. and B.V. Raushenbakh, "Automatic Assembly in Space." NASA Translation TT F-12, 113 presented at 19th Congress of the International Astronautics Federation, 1968.

3. Bergmann, E.V., et. al., "An Advanced Spacecraft Autopilot Concept," Journal of Guidance and Control, May-June 1979.

4. Simmons, R., E. Bergmann, B. Persson, W. Hollister. "Six Dimensional Trajectory solver for Autonomous Proximity Operations." AIAA paper 90-3459, presented at AIAA Guidance, Navigation, and Control Conference, Aug 1990.

5. McInnes, C.R., "Autonomous Proximity Manoeuvering Using Artificial Potential Functions," *ESA Journal*, Vol. 17, 1993, No. 2: 159-169.

6. Raquet, J. F. "Six Degree of Freedom Trajectory Planner for Spacecraft Proximity Operations Using an A* Node Search." Master of Science Thesis, M.I.T., 1991.

7. Niya, Craig K. "An application of the A* Search Technique to Trajectory Optimization." Master of Science Thesis, M.I.T., 1990.

8. "Model for Predicting Orbiter PRCS Plume Impingement Loads," Johnson Space Center Memo JSC-26507,. NASA Johnson Space Center, 1993.

9. Weiler, P. "Initial Exploration of Dynamic Plume Avoidance," CSDL Memo No. CC-88-03, Charles Stark Draper Laboratory, June 1988.

10. Vaughn, R.M., E.V. Bergmann, and B.K. Walker "Collision Detection for Spacecraft Proximity Operations." *Journal of Guidance, Control, and Dynamics,* Mar-Apr 1991: 225-229.

11. Pearl, Judea. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Reading: Addison-Wesley Publishing Complany, 1984.

12 "Functional Subsystem Software Requirements, Guidance, Navigation and Control, Part C: Flight Control, Orbit DAP," NASA Document #STS 83-009F, OI-23, February, 1990.

13. Horn, Berthold K.P., "Closed-form Solution of Absolute Orientation Using Unit Quaternions." *Journal of the Optical Society of America,* April 1987: 629-642.

14. Wertz, James R., ed. *Spacecraft Attitude Determination and Control.* Boston: D. Reidel Publishing Company, 1978.

15. Graham, Ronald E., "Contribution of a Rigid Body Dynamic Model ro Plume Impingement Analysis for the Space Station Freedom," *First IEE Conference on Control Applications,* IEEE, 1992: 47-51.

16. Roux, J.A. and T.D. McCay, eds. *Spacecraft Contamination: Sources and Prevention*. New York: American Institute of Aeronautics and Astronautics, Inc., 1984.

18 Pohl, I., "Practical and Theoretical considerations in Heuristic Competence, Genuine Dynamic Weighting and Computational Issues in Heuristic Problem Solving." *Proc. IJCAI 3*, Stanford, 1973: 20-23.

19. Press, W. H., Teukolsky, S. A., Vettering, W. T., Flannery, B. P., *Numerical Recipes in C.* pp336-338. Cambridge: Cambridge University Press, 1992.